UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior - Leganés

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Analysis of SoftToken: a Coordinated Medium Access Control (MAC) for IEEE 802.11 based Wireless Mesh Networks

AUTOR: LUCAS EZNARRIAGA BARRANCO
TUTOR: DR. JOSÉ IGNACIO MORENO NOVELLA

Leganés, 2010

TÍTULO: *Analysis of SoftToken: a Coordinated Medium Access Control (MAC) for IEEE 802.11 based Wireless Mesh Networks.*

AUTOR: Lucas Eznarriaga Barranco {lucas.eznarriaga@imdea.org}

TUTOR: Dr. José Ignacio Moreno Novella {joseignacio.moreno@ieee.org}

La defensa del presente Proyecto Fin de Carrera se realizó el día 4 de Octubre de 2010; siendo calificada por el siguiente tribunal:

PRESIDENTE: Andrés Marín López

SECRETARIO: Pablo Serrano Yáñez-Mingot

VOCAL: Francisco Javier González Serrano

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN:

Presidente **Secretario** **Vocal**

Acknowledgements

This thesis would have not been possible without the help and the support of many people.
I would like to thank my coordinators both in Madrid and Berlin, José Ignacio Moreno Novella and Hans J. Einsiedler for making the Internship at the Deutsche Telekom Laboratories possible under the Erasmus Placement programme, which has given me the opportunity of participating in the European Union 7th Framework Programme project CARMEN.
I would also like to thank my supervisors, Cigdem Sengul and Nico Bayer, for guiding me into the world of research.
Thanks to my friends for making me feel at home not only in Ibiza but also in Madrid or Berlin.
Last but not least, my deepest gratitude to my family and other animals...

><(((º> ><(((º> ><(((º>

Abstract

Wireless Mesh Networks (WMNs) which can be integrated to other types of networks at a reasonable cost are good candidates to facilitate the efficient and flexible deployment of Next Generation Networks (NGNs).
Due to the widespread presence of IEEE 802.11 Wireless Local Area Networks (WLANs) most of current WMNs are based on this technology. However, IEEE 802.11 is primarly designed for one-hop networks and its random access protocol (CSMA/CA) is problematic in multi-hop environments due to collisions, which decrease efficiency and service quality. To alleviate these problems a novel approach, SoftToken, was proposed to provide a coordinated Medium Access Control (MAC) for WLANs whose goal is to add a token-passing mechanism on top of the standard IEEE 802.11 in a way that collisions can be avoided. In this thesis, we present an analysis and evaluation of SoftToken in various scenarios. We also develop some functions to integrate this protocol into the architecture designed by the CARrier grade MEsh Networks (CARMEN) project.

Contents

List of Figures

List of Tables

Acronyms

3G	Third Generation
4G	Fourth Generation
AAA	Authentication, Authorization and Accounting
ACK	Acknowledgement
AI	Abstract Interface
AIFS	Arbitration IFS
AP	Access Point
BT	Backoff Time
BS	Base Station
BWA	Broadband Wireless Access
CARMEN	CARrier grade MEsh Networks
CAP	CARMEN Access Point
CGW	CARMEN Gateway
CHF	Capacity Handling Function
CMF	Capacity Management Function
CMP	CARMEN Mesh Point
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CW	Contention Window
DCF	Distributed Coordination Function
DVB	Digital Video Broadcasting
IFS	Interframe Space
IMF	Interface Management Functions
ISP	Internet Service Provider
LA	Link Agent
LG	Link Group
LGC	Link Group Controller
LL	Logical Link
LOS	Line of Sight
LTE	Long Term Evolution
MAC	Medium Access Control
MAd	MAC Adapter
MAN	Metropolitan Area Network
MANET	Mobile Ad Hoc Network
MBWA	Mobile Broadband Wireless Access
MeM	Measurement Module
MIH	Media Independent Handover

MIMO	Multiple Input Multiple Output
MMF	Mobility Management Function
MoMa	Monitoring Module aggregator
MoMs	Monitoring Module storage
MPLS	Multi-Protocol Label Switching
MSC	Message Sequence Chart
NGN	Next Generation Network
NGMN	Next Generation Mobile Network
NIC	Network Interface Card
PCF	Point Coordinated Function
PHY	Physical Layer
PLCP	Physical Layer Convergence Protocol
PMP	Point to Multipoint
QoS	Quality of Service
RtF	Routing Function
RTS/CTS	Request To Send / Clear To Send
SCF	Self-Configuration Function
SDR	Software Defined Radio
SS	Subscriber Station
STA	Station
TC	Traffic Class
TDDF	Topology Discovery and Dissemination Function
TDMA	Time Division Multiple Access
TETRA	TErrestrial Trunked RAdio
T-Labs	Deutsche Telekom Laboratories
ToS	Type of Service
UT	User Terminal
VoIP	Voice over IP
WiFi	Wireless Fidelity
WiMAX	World Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WMN	Wireless Mesh Network
WP	Work Package
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter 1

Introduction

The object of this thesis is to present an analysis and evaluation of SoftToken which was performed as an Erasmus Placement intern at the Seamless Communications department of Deutsche Telekom Laboratories (T-Labs) in Berlin from February to September 2010.
In this first chapter we provide an overview of this document. Firstly we present the technical motivation and the goals, then we explain the structure of the thesis.

1.1 Motivation

Over the last decades, wireless communications have experienced tremendous advances, which have transformed our everyday lives. Wireless access technologies such as cellular systems (3G, 4G), IEEE 802.11 WiFi (Wireless Fidelity) or IEEE 802.16 WiMAX (World Interoperability for Microwave Access) are evolving to offer mobile and ubiquitous broadband access to wireless users. On the one hand, 3G (third generation) cellular systems, which already provided worldwide coverage, evolve into 4G (fourth generation) of cellular networks such as LTE (Long Term Evolution) to provide peak rates exceeding 300 Mbps in the downlink and 75 Mbps in the uplink [8]. On the other hand, WiFi, which is widely deployed, is about to roll out the emerging IEEE 802.11n standard to provide throughput values up to 540 Mbps. WiMAX Wireless Metropolitan Area Networks (MANs) in turn, are expected to be increasingly deployed in the next years to provide Broadband Wireless Access (BWA) in urban areas [9]. Recently, Wireless Personal Area Networks (WPANs) as well as Wireless Sensor Networks (WSN), which are short-range, low-power and low-cost networks have also started to play a significant role in many different areas such as wireless control and monitoring, or home automation.
The integration of this heterogeneous networks is needed to provide seamless availability of data, voice and multimedia traffic [10]. In this context, the Next Generation Mobile Networks (NGMNs) represent the architectural evolution towards an all-IP packet based of both

telecommunication core and access networks making use of multiple broadband and QoS-enabled transport technologies independently of the underlying transport technology [11].
Wireless Mesh Networks (WMNs) which can be integrated to other types of networks at a reasonable cost are good candidates to facilitate the efficient and flexible deployment of Next Generation Networks (NGNs). However, the design of these networks represents a major number of research challenges such as scalability, QoS support or self-configuration. This constitutes the motivation for the CARMEN European project which aims to deliver carrier-grade services over heterogeneous WMNs at a reasonable cost. A whole architecture is designed and developed to allow heterogeneous WMNs consisting of multiple technologies. Most of WMNs are based on IEEE 802.11 which uses CSMA/CA that scales bad in multi-hop environments due to the collisions which decrease the efficiency. By adding a coordinated medium access mechanism on top of IEEE 802.11 it is possible to control the medium access avoiding collisions and increasing the channel efficiency. The main focus of this thesis is the analysis of SoftToken: a token-based protocol which has been designed for the CARMEN project and which implements a coordinated medium access for Wifi-based WMNs, avoiding collisions and enabling support for QoS.

1.2 Contributions

The main goals of this thesis are: evaluating, debugging and developing new functionalities for SoftToken.
The main contributions of my thesis are:

- Studying the performance of SoftToken both theoretically and practically and comparing it to the underlying IEEE 802.11 technology.
- During the evaluation phase some bugs were discovered in the implementation, which could affect the performance and therefore, needed to be solved.
- While debugging the code, tests were run for different scenarios and for different configurations to validate the models and find the operational bounds of SoftToken.
- We also implemented a new module for SoftToken integrating it into the CARMEN architecture by enabling the use of CARMEN primitives that set up SoftToken and perform resource management operations on it.
- We implemented a dynamic scheduler which enables the resource management primitives of CARMEN to modify the configuration of the scheduling parameters in SoftToken. These scheduling parameters determine the bandwidth and traffic class provided by SoftToken.

1.3 Structure

This thesis is structured as follows:

- **Chapter 2** presents the background regarding the terms and the standards that are related to the work presented in this thesis such as: WMNs, MAC technologies such as IEEE 802.11 and the token principle underlying SoftToken.
- **Chapter 3** describes the structure of the CARMEN European project. It presents the main objectives and its two example use cases. It introduces the functional components of the system and finally, the MAC mechanisms and the extensions that are supported, which is included in the scope of this thesis.
- **Chapter 4** presents the SoftToken Protocol, an IEEE 802.11 Coordinated MAC extension provided by the Deutsche Telekom Laboratories (T-Labs) for the CARMEN project, which is the subject of this thesis. First, the mechanisms of SoftToken along with its parameters and packet formats are introduced, then the implementation and the new functionalities necessary for the integration to the CARMEN project are explained.
- **Chapter 5** analyzes the performance of IEEE 802.11 in DCF mode and then, extending the same analysis, calculates the theoretical performance of SoftToken in a simple scenario.
- **Chapter 6** presents the results for the performance and validation tests for different configurations of SoftToken for both UDP and TCP. We always compare the performance of SoftToken with IEEE 802.11a, and for some of the tests, also with another software TDMA implementation. These tests expose the operational bounds of SoftToken.
- **Chapter 7** presents the conclusion and a summary of the results as well as some suggestions for future work.
- **Appendix A** presents the budget of the project.

Chapter 2

Wireless Mesh Networks (WMNs)

In this chapter we present the technologies and standards related to the Wireless Mesh Networks. We first introduce WMNs, which constitutes the core of the CARMEN project. We also present the characteristics of the MAC protocols for mesh networks and provide a summary of the available standards. Specifically, we introduce some previous token-based protocols such as IEEE 802.5 Token Ring and the more recent Wireless Token Ring protocol which are related works to SoftToken.

2.1 Overview

Wireless Mesh Networks (WMNs) are proposed to form a wireless backbone to provide multihop wireless connectivity to the clients [12]. WMNs constitute a networking architecture which is the result of the evolution of the existing wireless technologies such as IEEE 802.11 WLAN and IEEE 802.16 WiMAX whose aim is to provide better services at a reasonable cost. For this reason, it is considered by operators, Internet service providers (ISPs), governments and businesses as an attractive solution to deliver wireless broadband services with a minimal investment [18].
The history of WMNs has its origins in Mobile Ad Hoc Networks (MANETs). In the early 1980s the Survivable Adaptive Radio Networks (SURAN) program was created by the US Department of Defense (DoD) to provide packet switched networking to mobile battlefield elements in an infrastructureless, hostile environment, where soldiers, tanks or aircraft formed the nodes in the network [13]. The difference between MANETs and WMNs is that the first is formed by mobile nodes which result in dynamic topologies changing quickly and randomly and whose main goal is to provide networking in any environment while the second allows mobile as well as fixed wireless nodes with no energy constraints that form an infrastructure and whose aim is to provide ubiquitous access, high throughput and high data rate [14].
Although WMNs are relatively easy to deploy due to the fact that the components which form them are off-the-shelf, the existing MACs and routing protocols do not have yet the scalability

nor the throughput to deliver the desired broadband services. This constitutes one of the motivations for the CARMEN project, which represents the framework for this thesis and that will be explained in the next chapter.

2.1.1 Architecture of WMNs

Adopting the classification proposed by [15], nodes in a WMN can be mesh routers or mesh clients. The basic property of a WMN is that every mesh router but not necessarily every mesh client can act as a router forwarding packets to another node without the need of a common access point. The difference between the two types of nodes is that mesh routers form the backbone of the WMN and can have either gateway or bridge function. Additionally, they are equipped with multiple wired and wireless interfaces, which can possibly be of heterogeneous technologies. Mesh clients in contrast, are simpler: they do not have gateway capabilites nor multiple wireles interfaces, they have limited memory and computational power but still can communicate with each other forming client WMNs. Hence, there are three types of WMNs:

- Infrastructure/backbone WMNs are self-healing, self-configuring WMNs formed by mesh routers providing connectivity to the mesh clients as well as the integration of other networks through gateways.
- Client WMNs are formed by mesh clients.
- Hybrid WMNs are formed by the combined architecture of both previous types of WMNs.

Figure 2.1 shows an example of a Hybrid WMN containing both an Infrastructure WMN and a Client WMN.

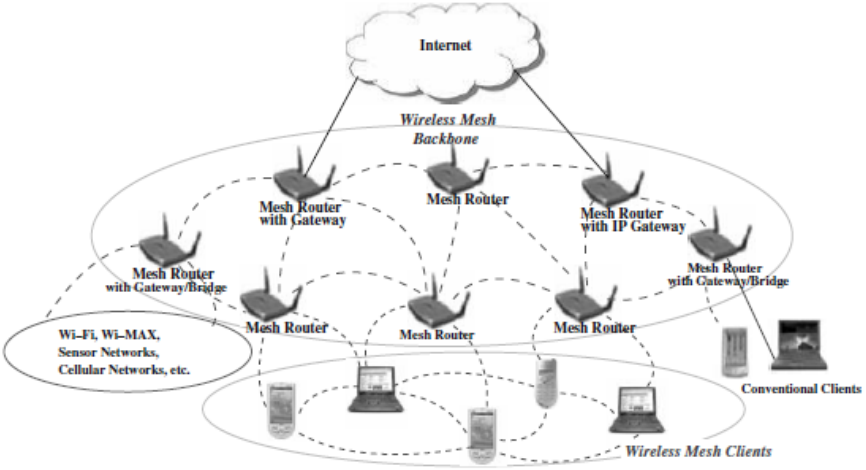


Figure 2.1: Example of a Hybrid WMN [1]

Some of the characteristics of WMNs include [12]:

- **Reliability** which is a consequence of the redundancy provided by the existence of multiple paths between nodes.
- **Low deployment costs** because of the availability of commercial off-the-shelf components and the lack of a wired infrastructure. As well as the possibility to operate in the unlicenced spectrum.
- **Large coverage area** which can extend the coverage of current wireless networks by multihop communication.
- **Self-configuration** and **Self-healing** capabilities that automatically establish and maintain network connectivity.

These characteristics make WMNs an effective solution in a large range of scenarios [15]:

- Broadband home networking as an alternative to IEEE 802.11 WLANs. Replacing the WLAN access points by wireless mesh routers with mesh connectivity, it is possible to achieve a more flexible and more robust communication.
- Neighbourhood networking in which a unique path can be established without the need to go through the Internet [16].
- Enterprise networking in which IEEE 802.11 WLAN access points are wired using Ethernet connections. As in the case of home networking, replacing the APs by wireless mesh routers, multiple connections to the backhaul can be shared by all nodes, improving the robustness and the resource utilization of the enterprise network [17].
- Urban as well as rural scenarios where the cost-effective deployment of WMNs can bring good coverage to in areas otherwise unaccessible to traditional broadband networks. These scenarios have gained the attention of network operators as well as Internet Service Providers (ISPs) as an attractive solution to provide broadband connectivity at a reasonable cost [18].
- Transportation systems providing wireless access inside trains, buses and ferries.
- Building automation for monitoring and controlling different devices without the need for a wired infrastructure.
- Health and medical systems for transmitting diagnosis and high resolution monitoring data.
- Security surveillance providing the capacity for continuous video streaming without the need for deploying any infrastructure.
- Emergency and disaster scenarios in which connectivity can be quickly re-established by means of a WMN.

The number of WMNs deployed or planned for deployment by local municipalities all around the world is too big to be enumerated here – only in US about 300 municipalities [19]. Examples of successful community mesh networks are the MIT Roofnet [20] or the Berlin Freifunk [21]. A list of wireless community networks by region can be found in [22]. Examples of research testbeds which have been developed to experimentally evaluate mesh limitations and capabilities are the Berlin Open Wireless Lab (BOWL) [23] which maintains a reconfigurable outdoor testbed with 50 nodes. A mesh testbed has also been deployed at the Deutsche Telekom Laboratories in Berlin for the final demo of the CARMEN Project [24] which will take place in January 2011. The mesh testbed for CARMEN is composed of 9 mesh nodes equipped with multiple interfaces of heterogeneous technologies.

2.1.2 Open Problems in WMNs

Current solutions for WMNs are far from being optimal. The lack of a single standard for WMNs creates a challenge as multiple technologies need to co-exist and requires the integration of them. The main design objective in WMNs is coverage and performance which are critically influenced by the following factors [12]:

- Advanced radio techniques such as reconfigurable and cognitive radios, Multiple Input Multiple Output (MIMO) systems, directional and smart antennas, multi-radio and multi-channel systems which have undergone a significant revolution but are still complex and expensive.
- Integration of multiple-technology networks which increase the service and network convergence and reduce the need for new infrastructures.
- Scalability which is a major issue due to the wireless and multihop nature of WMNs. As the number of nodes and hops increases, contention at the MAC level increases therefore decreasing the performance. Routing protocols may not find a reliable path and transport layer protocols may lose connections.
- QoS and service provisioning to support data, voice and TV (Triple Play) services have strict requirements in terms of end-to-end delay, jitter, packet loss and aggregate and per-node throughput which need to be respected in WMNs.
- Self-configuration and topology control are critical to provide mesh connectivity as well as to reduce the deployment and maintainance costs.
- Mobility support for mobile clients, who require efficient, low latency handover mechanisms.
- Monitoring and management tools are needed to keep track of the capacity and the network performance and to maintain the network operation.
- Security schemes and mechanisms for encryption, authorization, authentication, public distribution or intrusion detection specially designed for WMNs are necessary to deliver reliable services.

CARMEN tries to address the majority of these issues with its new architecture to integrate heterogeneous technologies.

2.2 MAC Layer in WMNs

MAC layer protocols coordinate the process of sharing the wireless medium among neighbouring stations providing a certain QoS in the physical and link layers in terms of throughput, delay, jitter or packet loss. There are two families of MAC protocols depending on the coordination of the access to the medium:

- Reservation-based MAC protocols where resources such as time slots, channels or bandwidth are assigned to users to satisfy throughput and QoS. Such MACs require cross-layer design to learn the QoS requirements, which otherwise, remain unknown to the MAC layer. Examples of this are TDMA MAC.
- Random access protocols such as Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA), whose main goal is minimizing collisions and fast recovery from them. However, the throughput performance decreases quickly as the number of nodes increases due to the transmission-collision-retransmission-collision patterns. This creates a scalability problem and makes QoS provisioning challenging.

MAC protocols are built over the physical layer and some of their components include: the packet processing and queuing for transmission and reception, the medium access and the network formation and association. In general, reservation-based MAC protocols are used in architectures which are centralized such as cellular networks or satellite networks. Alternatively, random MACs seem more suitable for wireless multihop networks such as WMNs which are intrinsically distributed.

There are currently three trends in implementation architectures for MAC protocols:

- The classical implementation where the protocol is implemented in software (MAC driver), firmware and hardware.
- Software MAC (softMAC) approaches where an additional software module or the driver can control and modify more functions of the MAC protocol. However, timing critical functions are hard to modify. In this thesis, we analyse and develop some functions for a software MAC called SoftToken which has been developed for the CARMEN WMN.
- The software defined radio (SDR) MAC architecture which constitutes a novel approach in which all timing critical functions can be modified by the driver.

2.2.1 Problems of IEEE 802.11-based MAC protocols

As mentioned before, a distributed MAC such as CSMA/CA is prefered for WMNs. However, although the ad hoc mode of IEEE 802.11 can be used to form a meshed wireless LAN, many modifications are needed to improve its performance for WMNs. Even after these modifications, the scalability problem remains. Some of these modifications include [1]:

- Tune physical carrier sensing to find a trade off between the hidden node and the exposed node problems by using dynamic carrier sense mechanisms.
- Improving virtual carrier sensing – RTS/CTS like – mechanisms to also find a trade off between the hidden node and the exposed node problems.
- Improving the backoff mechanism by dynamically tuning the contention window.

Instead of just fine tuning the parameters of CSMA/CA to improve its performance in WMNs, new MAC architectures are proposed to be integrated with CSMA/CA. This is the approach followed in CARMEN for the IEEE 802.11 based coordinated MACs that will be explained in the next chapter.

To improve multi-hop mesh Mac access, IEEE 802 standards committee is currently promoting a number of MAC standards related with WMN technologies through several working groups. Some of these standards are:

- 802.11e [25] is an approved amendment to the IEEE 802.11 standard which extends CSMA/CA to support different QoS traffic classes by defining a new mode of operation: the hybrid coordination function (HCF) and four access categories with different priorities.
- 802.11s [26] specifies both routing and MAC protocols as well as most of the design objectives described in section 2.1.2 for WMNs based on 802.11. It also adds support for broadcast and unicast services. However, many issues have not yet been addressed and are still object of discussion.
- 802.16a standard includes a mesh mode where direct communications among client nodes are also possible.

2.2.2 Token-based protocols

The idea of coordinating the access to a common shared medium by assigning transmission turns to computers using a packet called token can be traced back to the 1970s. At that time, IBM developed Token Ring used for Local Area Networks (LANs) where stations were connected in a ring topology that was later standardized by IEEE standard 802.5 [27]. In Token Ring, the token is passed in one direction around the ring and the computer that holds the token is allowed to transmit. After transmitting, it puts the token back on the ring. The standard defines data rates of 4 or 16 Mbps. Its major drawback is that it is very sensitive to faults in the ring which can disable the whole network.
Another protocol implementing a token passing mechanism for LANs is Token bus standardized by IEEE standard 802.4 [27] where the token is passed from one node to the next in a virtual ring. The advantage in respect to Token Ring is that it has the reliability and low cost of a bus network.
The main features of the token-passing networks can be summarized as:

- There are no collisions, as every station must wait for the token to transmit.
- They are deterministic in the sense that it is possible to calculate how long a station will wait to transmit.

These characteristics make token-based networks ideal for applications in which delay must be predictable and robust network operation is important [28] – e.g. QoS-sensitive applications. The war of standards for LANs between Ethernet, Token bus and Token Ring was won by Ethernet, mostly because it was there first and the challengers were not as good [29].
The token-passing mechanism has again gained the attention of the research community as a promising way to support QoS in wireless networks due to the unreliable nature of the medium and the incapacity of the existing CSMA/CA MAC protocols such as IEEE 802.11 to efficiently support QoS.
In section 4, we present the analysis of performance of SoftToken which represents a novel approach for coordinated MAC in wireless networks. It is based on the token principle, and its goal is to add a token-passing mechanism on top of the standard IEEE 802.11 in a way that collisions can be avoided.
Other token-based protocols for MAC in wireless networks have been released in the last years, a good example is the Wireless Token Ring Protocol (WTRP) [30] which constitutes a distributed MAC protocol for ad hoc networks. WTRP implements the Token Ring protocol on top of the IEEE 82.11 in the Distributed Coordination Function (DCF) mode.

Chapter 3

The CARMEN Project

3.1 Introduction

In this chapter, we summarize the CARMEN project which is the framework for which this thesis has been developed.
The CARrier Grade MEsh Networks (CARMEN) project focuses on specifying and developing the architecture of a heterogenous technology Wireless Mesh Network (WMN) supporting carrier-grade services[2]. It is a three-year project that started in January 2008 which is partially funded by the European Union´s 7^{th} Framework Programme and is composed of eight European partners – two network operators: Deutsche Telekom and BT, two equipment manufacturers: NEC and Alcatel Lucent, one research institute: Fraunhofer FOKUS and three universities: University College Dublin, Universidad Carlos III de Madrid and AGH University of Science and Technology of Krakow.

3.2 Overview of the CARMEN Objectives

The main goal of CARMEN is to overcome the limitations of WMNs in throughput and scalability. Some of the CARMEN objectives include [2]:

- Providing carrier-grade services in WMNs by offering services with a quality as close as possible to wired carrier-grade ones.
- Support for multiple radio access technologies by designing an interface which provides an abstraction of different radio-based MAC layers for WMNs such as IEEE 802.11, IEEE 802.16 or Digital Video Broadcasting (DVB).

- Development of extensions of the current IEEE 802.11 and IEEE 802.16 Medium Access Control (MAC) technologies to improve their efficiency in WMNs.
- Efficient usage of radio resources enhancing MAC layer functionalities and to provide dynamic radio planning in addition to the traditional fixed radio design.
- Mobility support by adopting and extending the IEEE 802.21 framework which describes the architecture for Media Independent Handover (MIH). In this way, handovers among heterogeneous access technologies, and building and managing heterogeneous mesh networks are provided.
- Providing broadcast and multicast services such as DVB or mobile TV.
- Support for self-configuration and monitoring in order to reduce operation and management costs and to support dynamic topology changes.

3.3 Architecture

The CARMEN architecture has been designed to meet the demands of two scenarios which represent two very different cases of the functionality and might need to scale up to 100 km^2. First we will analyze these scenarios to outline the functional requirements to be fulfilled and then we explain the basic design of the CARMEN network and discuss how these requirements are met.

3.3.1 Urban Scenario

In this section we identify some of the characteristics of a WMN deployed in a metropolitan area. Some of the issues that need to be addressed in such a scenario include [2]:

- The size of the metropolitan area to be covered.
- The user density per area and over the time.
- The provided bit rates for the different services: voice, video and DVB.
- The density of network elements necessary to support the user demand.
- The technologies employed for both the hardware and the functionalities software.
- The type and density of wireless and wired connections. In WMNs, communication mostly exists between neighbouring nodes and wired connections between the mesh network and the operator's fixed infrastructure.
- The type of user traffic and its constraints.

- The type of signalling traffic such as management, configuration functions and mobility support.

Taking into account the needs of the users, the services that need to be offered by the WMN are: basic Internet access, support for voice calls, mobility support and support for broadcasting services.

3.3.2 Emergency Scenario

In this section, we identify some of the characteristics where multiple technologies might need to be deployed in an emergency scenario. In this scenario, WMN has the major advantage of providing a quick deployment to connect the area of the disaster with the existing communication network, which is assumed to be partially available.
Some of the characteristics include [2]:

- Communication to handhelds supporting both voice and data via Point-to-Point or Point-to-Multipoint.
- Support for multiple technologies such as TETRA, WiMAX, WLAN, DVB and satellite communication.
- Low data rates (e.g. TETRA) as well as high data rates up to 54 Mbps (e.g. WiFi) towards the end user and up to 100 Mbps (e.g. WiMAX) in the mesh backbone.
- Wired or wireless terrestrial gateway to the Internet.
- Satellite-based gateway to Internet.
- Coverage-oriented deployment in the beginning and increasing the capacity in later phases.
- Support for security mechanisms such as encryption.
- Traffic priorization and support for different Quality of Service (QoS) levels to distinguish control traffic from other data traffic classes.
- Self-configuration capabilities.

3.3.3 CARMEN Network

An example of a typical CARMEN topology of several wireless hops and mesh links of multiple wireless technologies is shown in Figure 3.1 where radio connectivity is shown in grey dotted lines. A CARMEN Mesh is a network formed by nodes connected via radio interfaces which provide access to a core network. End-user terminals are not considered as part of the mesh [2]:

1. User Terminal (UT) is a fixed or mobile end-user device which uses the CARMEN Mesh to acccess to the services.

2. CARMEN Mesh Point (CMP) is a CARMEN Mesh node equipped with capabilities such as traffic forwarding and traffic monitoring. It can have multiple radio interfaces.

3. CARMEN Gateway (CGW) is a CMP providing connectivity to the network provider´s core or backbone network.

4. CARMEN Access Point (CAP) is a CMP providing connectivity to the CARMEN Mesh to the UTs.

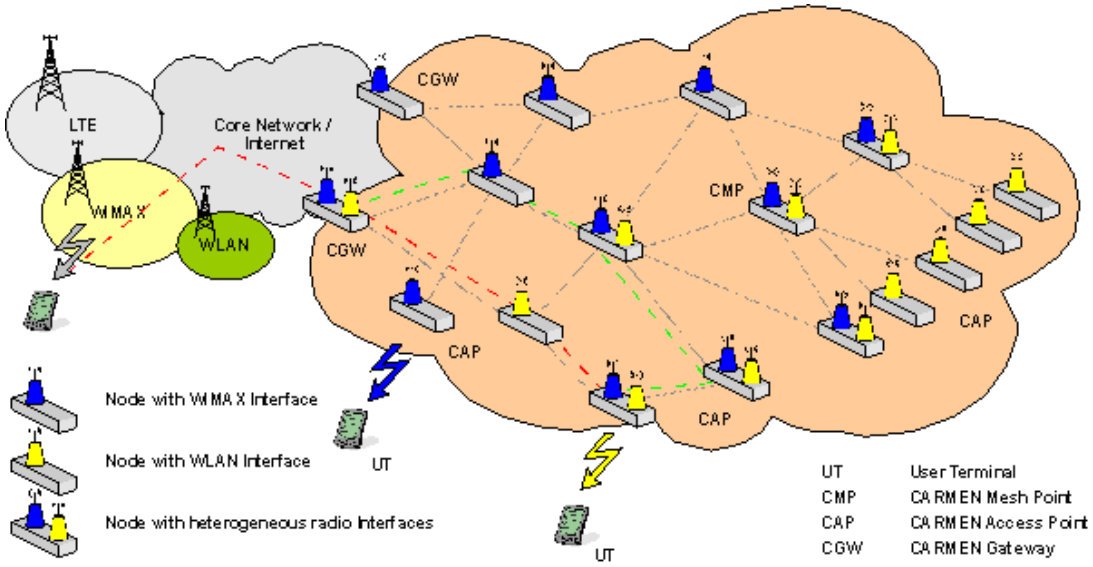


Figure 3.1: Example of CARMEN Network Topology [2]

In Figure 3.1 a connection is established for a CARMEN UT and a WiMAX UT to access to the Internet. Within the CARMEN Mesh, we can see load balancing among two different paths between the CAP to which the CARMEN UT is connected and the CGW which provides connectivity to the Internet.

3.4 Heterogeneous WMNs: CARMEN Technology Abstraction

The CARMEN technology abstraction aims to hide the underlying technologies providing only generic performance properties for links. In order to achieve this, CARMEN classifies different type of links [31].

- Physical Link is a feasible wireless communication channel between two or more nodes.

- Logical Link (LL) is a configured unidirectional channel between nodes which enables layer 2 communication making CARMEN Mesh communication possible.
- Link Group (LG) is a set of LLs sharing the same physical resources – location, area or spectrum. In a LG, there is a unique node called Link Group Manager (LGM) which is responsible for the management of a LG.
- Pipe is a technology-independent aggregate of traffic over multiple LLs, on which resource allocations have been performed and QoS is to be provided. Forwarding of the traffic aggregates will be based on Multi-Protocol Label Switching (MPLS).

Figure 3.2 shows an example setup where the nodes A, D and E form a LG where six unidirectional LLs have been configured by the LGC (node A). LL#1 for example, is established between the wireless interface 2 of node A and the wireless interface 0 of node E, and has two one-hop QoS allocations that can be of different traffic types.

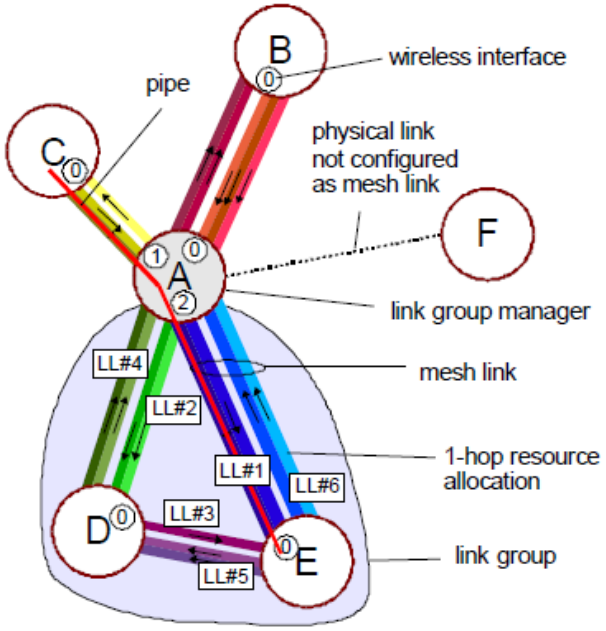


Figure 3.2: Example of a LG in a multihop scenario [3]

Since the LGCs are interface-specific they have to be implemented in a technology-specific way. In order to translate these technology-specific parts into a common set of primitives that can be used by the upper layer mesh domain functions in a technology-agnostic way, CARMEN defines two key components which are shown in Figure 3.3 [4]:

- The Interface Management Function (IMF) which uses the IEEE 802.21 primitives where possible and extends those with mesh specific requirements to be used by the upper layer mesh domain functions. Section 3.5.5 provides a more detailed description.

- The technology-specific MAC Adapters (MAds) which provide the resource abstraction and management, and comprise the LGM and the Monitoring Module aggregator (MoMa). We describe the CARMEN MAC extensions in detail in section 3.6.

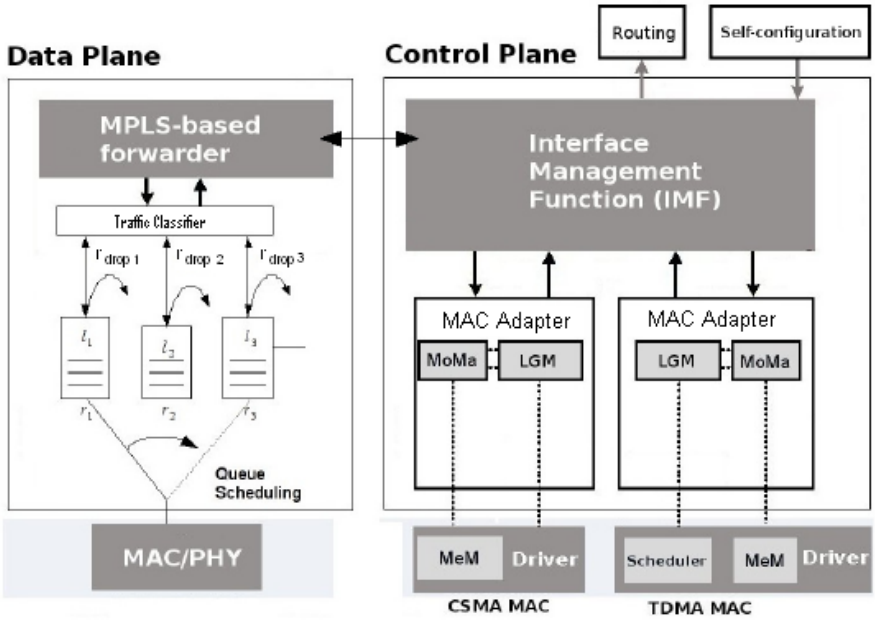


Figure 3.3: Data and control planes in CARMEN [4]

Figure 3.3 shows the data and control planes in a CARMEN node with two wireless interfaces. In the control plane, routing and self-configuration mesh functions operate on the abstraction layer, which is shown as the IMF in Figure 3.3. IMF provides a set of Abstract Interface primitives to the modules above and below it. The data plane depends on the underlying MAC technology and comprises the MPLS forwarder which is responsible for the traffic handling within a Pipe.
The modules and their functionalities defined by CARMEN which are part of the control plane are: self-configuration, monitoring system, capacity handling, routing, mobility management and the IMF. In the next section we describe briefly some of them.

3.5 Modules and Functional Requirements

3.5.1 Self-Configuration

The main goals of the Self-Configuration module are to provide: node discovery, initial topology formation, adaptation and optimization during regular operation as well as during network failure in an autonomous manner.

3.5.2 Monitoring System

The main goal of the Monitoring System module is to supply other modules accurate and timely information. It is formed by three submodules: Measurement Module (MeM), Monitoring Module aggregator (MoMa) and Monitoring Module storage (MoMs) whose main functions are as follows:

- MeM performs real-time measurement of several radio link parameters and neighbourhood scanning in a timescale of microseconds.
- MoMa performs statistical analysis of data and provides a smoothing functionality in timescale of seconds in order not to introduce over reactions.
- MoMs extends the MoMa functionality in a longer timescale so as to provide the selected measurements to the Self-Configuration module.

3.5.3 CARMEN Capacity Handling

All traffic in the CARMEN Mesh is classified in Traffic Classes (TCs). These allow differentiating packets with varying traffic forwarding requirements. Based on the traffic classes of IEEE 802.11e, CARMEN defines four TCs [2]:

- TC 1 – the highest priority TC – is for network control and management.
- TC 2 corresponds to first class guaranteed data rate service such as voice.
- TC 3 corresponds to second class guaranteed data rate service such as video.
- TC 4 corresponds to best-effort traffic.

Table 3.1 shows the correspondence with the traffic types defined in 802.11e.

TC	**Priority**	**Traffic Type**	**Correspondence with 802.11e**
1	1^{st}	Network ctrl and mngt	-
2	2^{nd}	First class guaranteed data rate service	VO
3	3^{rd}	Second class guaranteed data rate service	VI
4	4^{th}	Best-effort	BE, BK

Table 3.1: CARMEN Traffic Classes

The Capacity Handling Functions (CHFs) include:

- Admission Control mechanism checks whether a new allocation request for a certain traffic class and data rate to a given flow is allowed or denied.
- Pipe Initiation and Maintenance which, at the start-up of a CAP, requests least a pair of best-effort pipes to a CGW. It also monitors the data rates of the flows for a pipe are maintained.
- Policing Control Function and Policy Enforcement Function checks and enforces, respectively, that all subscriber flows are inline with their granted bandwidth.

3.5.4 Routing

The routing module provides connectivity between the CAPs and the CGWs, and manages the overall capacity within the CARMEN Mesh. On a network start-up, after the Self-Configuration, the Routing function (RtF) establishes a path to its nearest CGW and builds a logical link view of the network including QoS estimates for each link. The main functions of RtF include:

- Topology Discovery and Dissemination Function (TDDF) provides the mechanisms to build a global view of the mesh network at a logical link level.
- Capacity Management Function (CMF) is responsible for the setup, teardown and maintenance of pipes. It subscribes to the AI to receive 802.21 style MIH link events which inform the CMF when any change in the status of a local link has occurred, recomputing new routes and building new pipes if necessary.
- Forwarding Function computes the next hop in a per-pipe basis. Multi-path between CAPs and CGWs is supported using a label switching mechanism such as MPLS.

3.5.5 Interface Management Function

The IMF handles vertical message mapping and routing between modules above and below the Abstract Interface (AI) within a node which is the control plane service access point for manipulating links and LGs. The IMF provides a set of AI primitives to CARMEN higher layers such as RtF, SCF, CHF, MMF and MoMs as well as to lower layer MAds which translate the technology independent messages to technology dependent ones. IEEE 802.21 [32] defines media access independent mechanisms that enable the handover between heterogeneous IEEE 802 networks and facilitates handover between IEEE 802 and cellular networks namely Media Independent Handover (MIH). This characteristic is a main requirement for CARMEN which uses IEEE 802.21 primitives where possible and defines new ones for mesh specific functions.

3.5.5.1 Interface Specifications for the Resource Management Services

In this section we explain the AI service primitives which have been used later in this thesis. We will need them to set up a LG as well as to perform resource management on it. The first two primitives correspond to Link Information and Configuration Service primitives and are responsible for setting up a LG and the rest corresponds to the Resource Management Service primitives which perform the allocation, modification or release of the resources.

AI_Link_Set_Link_Group.request is used by the SCF to set the link group for a given link from the local IMF.
Semantics of the service primitive:

```
AI_Link_Set_Group.request (
NodeID,
RadioID,
LinkGroupID
)
```

Where:

- NodeID is the identifier of the node
- RadioID is the identifier of the radio interface
- LinkGroupID is the identifier of the link group

AI_Link_Set_Link_Group.response is used by the IMF to report back the result of setting the link group information to the requester.

```
AI_Link_Get_Link_Group.response (
Status
)
```

Where:

- Status is the result of setting the link group that can be success or failure

AI_Link_Allocate_Resources.request is used by RtF to allocate resources of an outgoing link to a given pipe. It is generated by the RtF on every node along a pipe's path during pipe setup to install resource allocation state. The IMF allocates resources for the specified pipe on the specified link and replies immediately with an *AI_Link_Allocate_Resources.response* primitive.
Semantics of the service primitive:

```
AI_Link_Allocate_Resources.request (
NodeID,
LinkID,
PipeID,
TrafficClass,
Rate
)
```

Where:

- NodeID is the identifier of the node
- LinkID is the identifier of the logical link on which resources are allocated
- PipeID is the pipe identifier for which the resources are allocated
- TrafficClass is the Traffic Class for traffic on this pipe
- Rate is the requested data rate

AI_Link_Allocate_Resources.response is used by the IMF to signal back the resource allocation success to the RtF.
Semantics of the service primitive:

```
AI_Link_Allocate_Resources.response (
Status
)
```

Where:

- Status is the result of the allocation attempt that can be success or failure

AI_Link_Modify_Resources.request is used by RtF to modify an existing allocation for a given pipe. It is generated by the RtF on every node along a pipe's path during pipe modification. The IMF reallocates resources for the specified pipe on the specified link and replies immediately with an *AI_Link_Modify_Resources.response* primitive.
Semantics of the service primitive:

```
AI_Link_Modify_Resources.request (
NodeID,
LinkID,
PipeID,
NewRate
)
```

Where:

- NewRate is the new requested data rate

AI_Link_Modify_Resources.response is used by the IMF to signal back the resource allocation success to the RtF.
Semantics of the service primitive:

```
AI_Link_Modify_Resources.response (
Status
)
```

Where,

- Status is the result of the reallocation attempt that can be success or failure

AI_Link_Release_Resources.request is used by RtF to release an existing allocation for a given pipe. It is generated by the RtF on every node along a pipe's path during pipe teardown.The IMF removes all the pipe state for the specified pipe and replies immediately with an *AI_Link_Release_Resources.response* primitive.
Semantics of the service primitive:

```
AI_Link_Release_Resources.request (
NodeID,
LinkID,
PipeID,
)
```

AI_Link_Release_Resources.response is used by the IMF to signal back the resource allocation success to the RtF.
Semantics of the service primitive:

```
AI_Link_Release_Resources.response (
Status
)
```

Where:

- Status is the result of the release attempt that can be success or failure.

3.6 CARMEN MAC Extensions

In this section, we present a detailed description of the CARMEN MAd as well as the technology-specific coordinated MAC which is presented in this thesis. Other coordinated MAC approaches as well as uncoordinated ones are implemented and evaluated in CARMEN, however, they are out of the scope of this thesis.

3.6.1 CARMEN MAC Adapter (MAd)

The MAds provide the mechanisms to ensure communication between the upper-layer technology-independent CARMEN modules and the underlying wireless links of heterogeneous technologies [5]. Communication with upper-layer modules is done by means of the AI primitives provided by the IMF. The MAd interfaces directly with the underlying MAC drivers of the wireless interface under its management. For this reason, in every CARMEN node, there is one MAd per radio interface.
At the bootstrap phase, a MAd is responsible for setting the initial configuration of its radio interface as governed by the SCF. The MoMa which is part of the MAd and communicates with the MeM, provides the SCF with a list of the discovered neighbours. With this information, the Self-Configuration Function (SCF) can decide which nodes belong to the same LG and designate a unique Link Group Controller (LGC). The rest of the nodes become Link Agents (LAs).
The LGC act as the coordinator between the MAds of the LG, managing the requested 1-hop QoS allocations by configuring the scheduler in the MAC layer for Coordinated MAC technologies or the parameters of the access mechanism in case of Uncoordinated MAC. The LGC is also responsible for keeping track of the allocated resources within the LG. The architecture of a MAd inside a CARMEN node is shown in Figure 3.4.

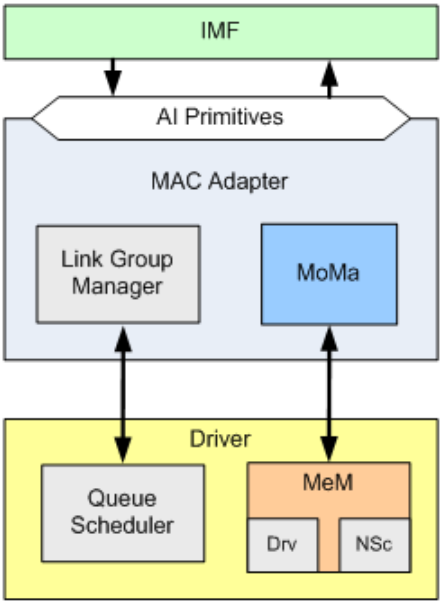


Figure 3.4: MAd inside a CARMEN node [5]

3.6.2 Coordinated MAC Technologies

CARMEN considers two different coordinated MAC technologies: WiMAX and IEEE 802.11-based coordinated MAC which adds a coordinated MAC mechanism on top of the IEEE 802.11 standard. In this thesis, we describe in detail SoftToken which is one of the two IEEE 802.11 coordinated MAC approaches developed by the T-Labs for the CARMEN Project.
Most of current WMNs are based on IEEE 802.11 technology. As it uses an uncoordinated MAC technology, due to its random access (CSMA/CA), in multihop networks, it might cause severe collisions which decrease the efficiency and the service quality.
A promising solution is adding a coordinated MAC mechanism on top of IEEE 802.11. In that way, it is possible to provide carrier-grade services because the access to the medium can be controlled so that no collisions occur and the resources can be explicitly reserved, giving support for QoS.
However, in coordinated MAC technologies the scheduling mechanisms are critical to provide carrier-grade support. Some of the issues that need to be addressed are:

- Transmission timing.
- Priority queues.
- Efficient resource assignment.
- Distributed, centralized and hybrid resource assignment.

The two coordinated MAC extensions which are being developed by TLabs for the CARMEN project are:

- SoftTDMAC over IEEE 802.11
- SoftToken over IEEE 802.11

SoftTDMAC over IEEE 802.11 is a software-based multihop Time Division Multiple Access (TDMA) MAC protocol based on the implementation by Petar Djukic [33]. It uses microsecond sized TDMA slots that make it very efficient and provide predictable transmission times. However, tight synchronization is needed.
The SoftToken over IEEE 802.11 implementation is the approach that is presented in this thesis and constitutes a software-based token protocol for wireless networks. The coordination for transmission is based on the token principle which avoids collisions and removes the need for synchronization. It can be seen as a TDMA system but with very relaxed slot borders; however, overhead is introduced by the token passing mechanism. In the next chapters we provide a detailed description of the SoftToken protocol.

Chapter 4

SoftToken Protocol

SoftToken is a software-based token protocol for WLAN which aims to provide the IEEE 802.11 uncoordinated MAC protocol the ability to deliver carrier-grade services by removing the contention in the access to the medium. One of the advantages of the SoftToken mechanism is that it is designed specifically to fit into the CARMEN architecture [1].
In this thesis, we present the characterization of the SoftToken in terms of performance – saturation bandwidth for both TCP and UDP, jitter and packet loss – and the functionalities developed to integrate it inside the CARMEN framework.

4.1 Architecture

In this section, we give an overview of the SoftToken mechanism. After the bootstrapping phase in CARMEN, the Self-Configuration module determines which CMPs that form a LG: which become LAs, and which node behaves as the LGC. In this scenario, the scope of SoftToken is a single LG.

4.1.1 Main Idea

In SoftToken, which is based on the token principle, the LGC works as the master which is the central entity that coordinates the transmission of the LAs that work as the slaves. The master sends regularly a token request message to one slave at a time – currently in a round robin way but other strategies are possible – which represents the token passing from the master to a slave. The token request message signals the resource allocation determining the amount of traffic that can be sent by the slave. The slave replies with a token response message after its data transmission. This constitutes the token passing from the slave to the master. In the token

[1] The developer of SoftToken is Karl Thiel from T-Systems and the current stable release is the 0.1.0.

response message, the slave indicates how much data has been transmitted (expressed in bytes or packets) and the amount of data which is enqueued waiting to be transmitted.
After receiving a token response message, the master transmits the token request message to the next slave or serves its own request based on the current resource allocation.
This mechanism for three CMPs is explained using the example described in Figure 4.1 which shows three mesh nodes and a Message Sequence Chart (MSC) depicting the frame exchange between them.

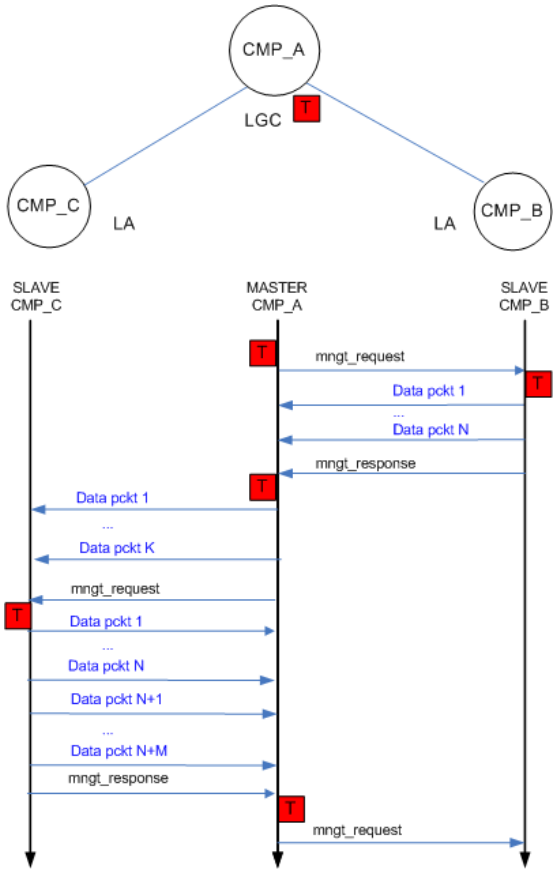


Figure 4.1: SoftToken mechanism example

In Figure 4.1, at the beginning, the node configured as the master owns the token which is represented by a read square marked with a T. Then it selects the next slave based on the current schedule – CMP_B – and generates the first management request packet. This mngt_request signals the scheduler configuration, which means for each transmission queue – current implementation of SoftToken defines 4 queues – the amount and the type of the data that can be transmitted by a slave before sending the management response back to the master. Every queue has an associated Traffic Class (TC) so there are as much as four TCs. Outgoing IP packets are queued by SoftToken in a queue with a corresponding TC according to the value indicated by the Type of Service (ToS) field of its IP header. As it will be explained later, this is used by the scheduler to allocate resources differently, permitting traffic differentiation and

enabling QoS.
Let us assume that the scheduler is configured as follows:

- in queue 0, N TC0 packets can be transmitted,
- in queue 1, K TC1 packets can be transmitted,
- in queue 2, M TC2 packets can be transmitted,
- in queue 3, 0 TC3 packets can be transmitted.

Then, when the slave CMP_B, which has packets of TC0, receives the mngt_request, it releases N packets from the queue zero and sends a token response to the master. During the time the token is at the slave CMP_B, data of TC1 has arrived to the master. However, although the configuration of the scheduler signals that K packets can be transmitted in queue one, the master has to wait for the token, the mngt_response, from CMP_B to release the K packets. Then the master selects the next slave, CMP_C, and sends the second mngt_request.
CMP_C has data enqueued in both queue zero and queue two. Then, when it receives the mngt_request from the master, it releases N packets from queue zero and M packets from queue two before transmitting the mngt_response back to the master.
If we compare this behaviour with the normal IEEE 802.11 we can see that collisions are avoided by SoftToken.

4.1.2 Packet Format

In this section, we explain the packet format of the messages used by SoftToken. SoftToken packets are called SoftToken management packets –*softtoken_mngt* –. Every *softtoken_mngt* packet is encapsulated into a UDP datagram. Figure 4.2 shows the encapsulation of a *softtoken_mngt* packet into a UDP datagram. SoftToken UDP datagrams can be of 50 bytes or 102 bytes.

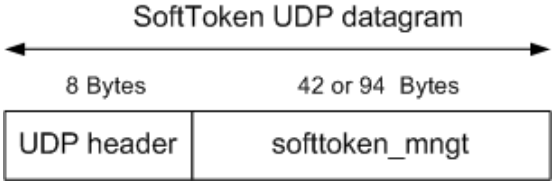


Figure 4.2: SoftToken management packet encapsulation in UDP

The fields of the UDP header are shown in Figure 4.3. Fields *source_port* and *dest_port* are filled with the SoftToken port, which is 434. The *checksum* is set to 0 and the *udp_length* is not used.

A *softtoken_mngt* packet has a length of 42 bytes if it carries a SoftToken management request – *softtoken_mngt_request* – or 94 bytes if it carries a SoftToken management response

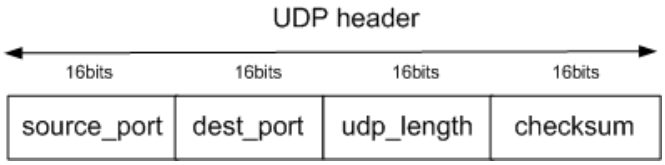


Figure 4.3: UDP header fields

– *softtoken_mngt_response* –. All *softtoken_mngt* packets carry a *softtoken_mngt_base* of 14 bytes which indicate the type of the *softtoken_mngt*. This is shown in Figure 4.4. The *softto-*

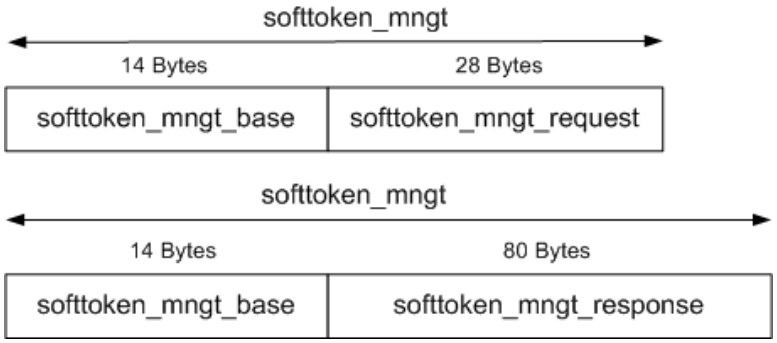


Figure 4.4: SoftToken management packet encapsulation

ken_mngt_base fields are shown in Figure 4.5.

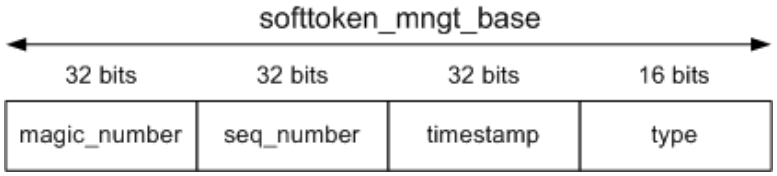


Figure 4.5: Fields of a *sofftoken_mngt_base*

The *magic_number* field is used to check if the packet belongs to the SoftToken protocol. The *seq_number* identifies a request - response cycle. It increases with each *softtoken_mngt_request* and is the same for the corresponding *softtoken_mngt_response*. The field timestamp is currently not used and set to 0. The field type is equal to 64 if the *softtoken_mngt* carries a *softtoken_mngt_request* and equal to 65 if it carries a *softtoken_mngt_response*.

The *softtoken_mngt_request* fields are shown in Figure 4.6. The *softtoken_mngt_request* is used to signal the resource allocation contained in the scheduler as well as to pass the token to a slave.

Each field *queue* contains the number of the queue to which the resource allocation refers. In the current implementation of SoftToken there are 4 queues. Each field *type* indicates what the

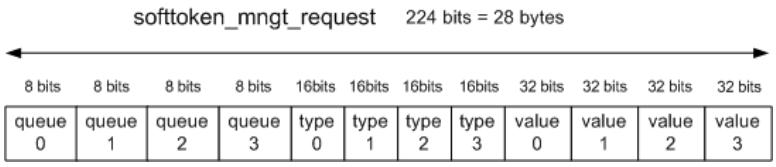


Figure 4.6: Fields of a *softtoken_mngt_request*

corresponding queue is carrying: 0 for packets, 1 for bytes and 2 for time, the last one, not implemented. Each field *value* indicates the value of the resource allocation for the corresponding queue and of the corresponding type. The *softtoken_mngt_response* fields are shown in Figure 4.7. The *softtoken_mngt_response* signals how much data has been transmitted and corresponds to passing the token back. It also signals the amount of data at each queue in the node.

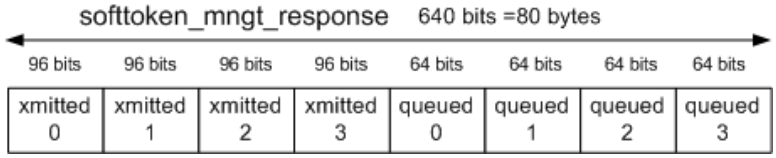


Figure 4.7: Fields of a *softtoken_mngt_response*

For each of the 4 queues, *softtoken_mngt_response* has a field *xmitted* and a field *queued* which are shown in Figure 4.8. As it has been explained, the *xmitted* field indicates the amount of data which has been transmitted in terms of either the number of packets , the number of transmitted bytes or the amount of time, the last one not implemented. And the *queued* field in turn indicates for each queue the data enqueued in a node in terms of number of packets which occupy a total size of bytes.

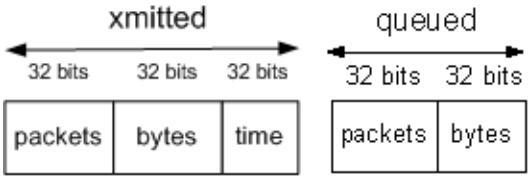


Figure 4.8: Subfields of every *xmitted* and *queued* field

4.2 Implementation

SoftToken has both a kernel and user space implementation as it is shown in Figure 4.9[34]. In user space, the SoftToken Controller module is used as a configuration and monitoring interface for the kernel space.
Any mesh node in a LG can eventually become the LGC (see section 3.6.1). To match this constraint, the SoftToken implementation is designed to work either in the master or the client (slave) mode.

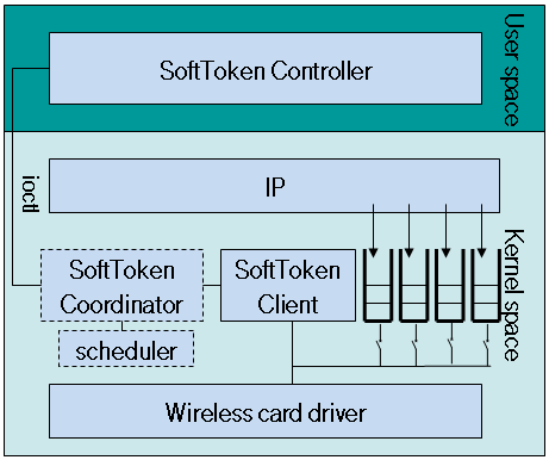


Figure 4.9: SoftToken architecture [6]

4.2.1 Kernel Space Implementation

The main modules in the kernel space implementation are the SoftToken Client, the SoftToken Coordinator and the scheduler. All the nodes running SoftToken in a LG have the SoftToken Client functionality, however, only the node working as the master, the LGC, additionally has the SoftToken Coordinator functionality. We start with explaining the SoftToken Client which is simpler, then we introduce the Coordinator.

4.2.1.1 SoftToken Client

The slave mode is the default mode of operation in SoftToken. The SoftToken Client module is implemented at each slave node along with four packet queues and uses the netfilter framework [35] to control the packet's traversal of the IPv4 protocol stack. It registers to two netfilter hooks:

- NF_IP_PRE_ROUTING which is the hook for the IPv4 incoming packets
- NF_IP_POST_ROUTING which is the hook for the IPv4 outgoing packets

When the SoftToken kernel module is inserted in slave mode, it implements only the client functionality: it registers the SoftToken device in the kernel, initializes some of the node's attributes and plugs into the Netfilter hook. After being inserted, a node running SoftToken still needs to be told by the SoftToken Controller which wireless interface is going to be used. These operations are shown in the flow diagram of Figure 4.10.

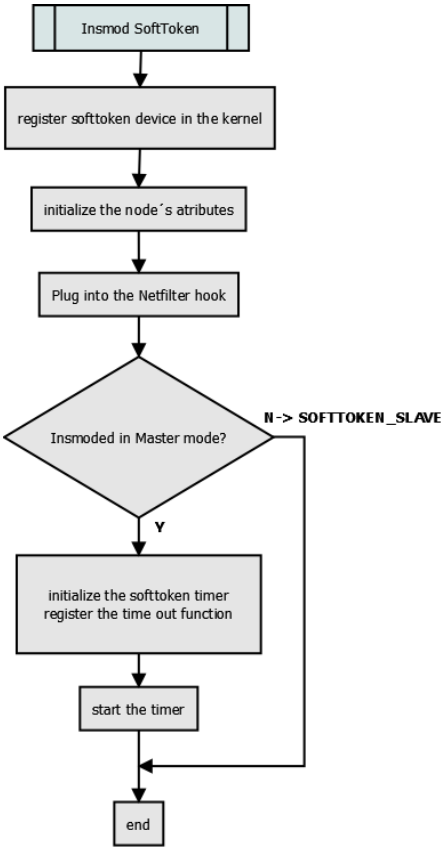


Figure 4.10: Flow diagram for SoftToken operations after the module is inserted

Once the wireless interface to be used has been configured, a slave node waits for a packet indication. The packet indication can occur because of an incoming or an outgoing packet. If an outgoing packet is destinated to the SoftToken interface, the ToS field of its IP header is examined and it is enqueued in the queue which has the same TC, otherwise it is enqueued in queue 0. If the packet is destined to another interface, it is allowed to pass through SoftToken. The flow diagram in Figure 4.11 shows this functionality.

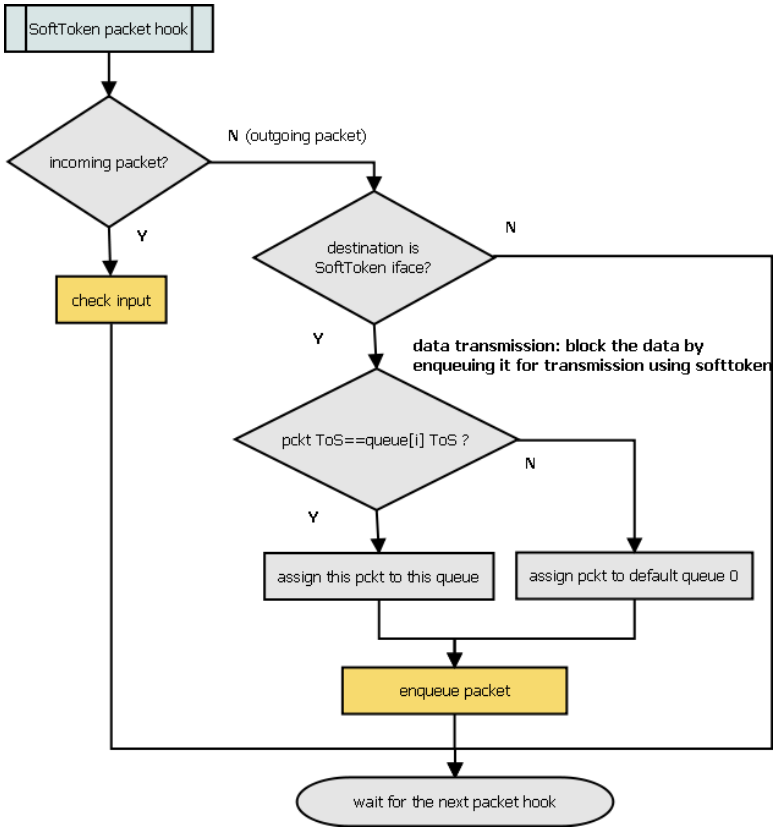


Figure 4.11: Flow diagram when a packet hook occurs

When an incoming packet is captured, the slave node needs to check whether the packet contains a SoftToken message from the master and then processes the message. If the message contains another type of packet, it is allowed to pass through SoftToken to the upper layers. Nodes in slave mode can only receive SoftToken management request messages from the master. So, when a *softtoken_mngt_request* is received, SoftToken releases from each queue as many packets or bytes as it has been indicated in the *softtoken_mngt_request* (see Figure 4.12). Then, it creates a SoftToken management response packet for the sender of the *softtoken_mngt_request* which is the master, sends it and waits for another packet hook to occur (see Figure 4.13).

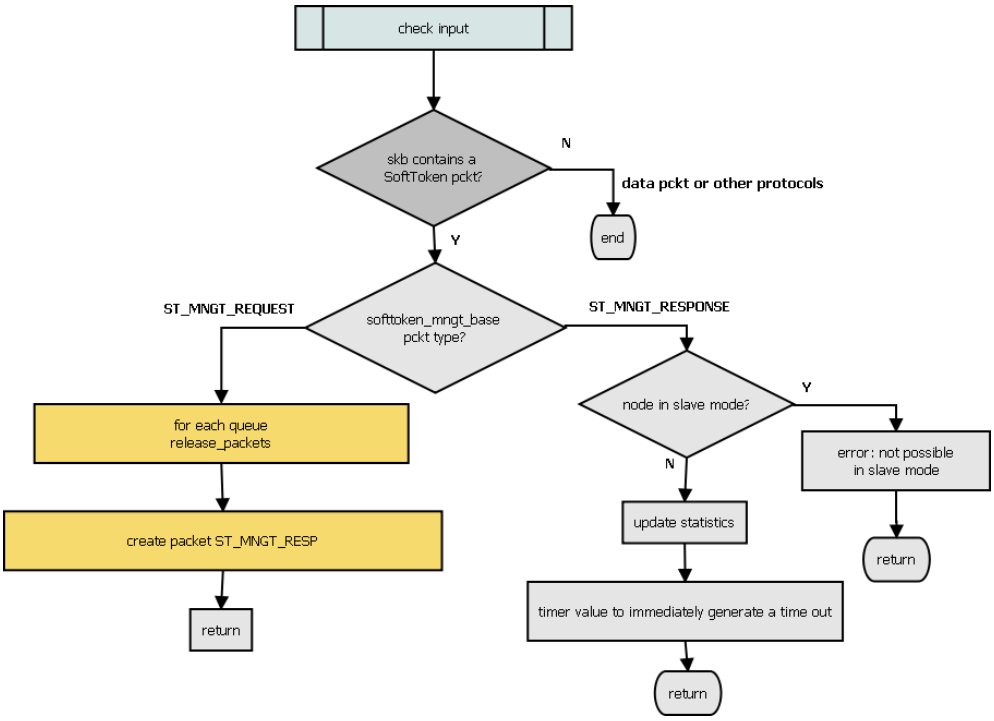


Figure 4.12: Check input functionality

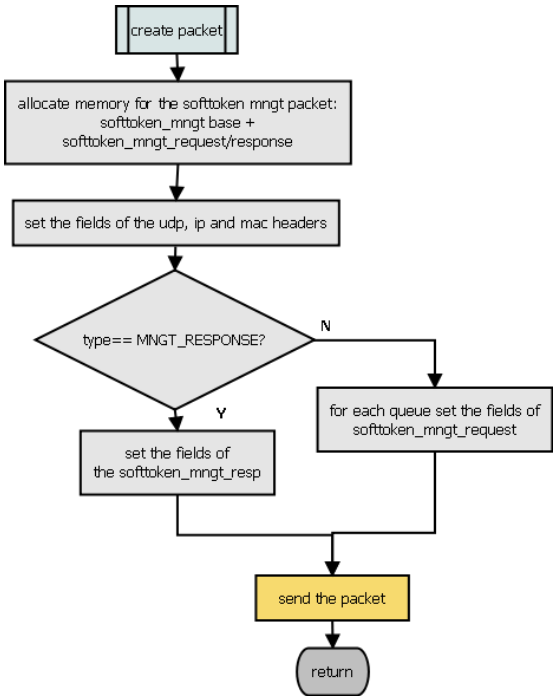


Figure 4.13: Create packet functionality of the kernel space

4.2.1.2 SoftToken Coordinator

The SoftToken Coordinator module is the responsible for the coordination of the token. For this reason it needs to add some functionality to the one already provided by the SoftToken Client module such as maintaining a list of the slaves running SoftToken, reliability mechanisms such as a retransmission timer to decide when the token has been lost and keeping track of the resource allocations. When the SoftToken kernel module is inserted in master mode, in addition to the client functionality, it initializes the SoftToken timer and registers the function to be called when a time out occurs (Figure 4.10).
Until a slave node is added to the Link Group (by using the SoftToken Controller to inform the master the MAC address of a new slave), the only node to be scheduled by the master is itself. Then, a time out occurs, the master schedules itself as the next node to request. It creates a *softtoken_mngt_request* for itself, sends it and sets the timer to 2 seconds. As the *softtoken_mngt_request* has been sent for the master itself, it produces a packet which is not transmitted by the wireless interface. Then, as it has been explained for the client functionality, it checks the type of the message, recognizes the *softtoken_mngt_request*, releases as many packets as it has enqueued according to self-signalled *softtoken_mngt_request* and creates a *sofftoken_mngt_response* to the sender of the *softtoken_mngt_request* which is again, the master itself. The *softtoken_mngt_response* in turn, produces another packet indication in the master which checks the input identifying the message. It updates the statistics for the packets that it has released and are still enqueued and sets the timer to produce a time out immediately, restarting the loop that has been explained in this paragraph. This is shown in Figure 4.14. When a

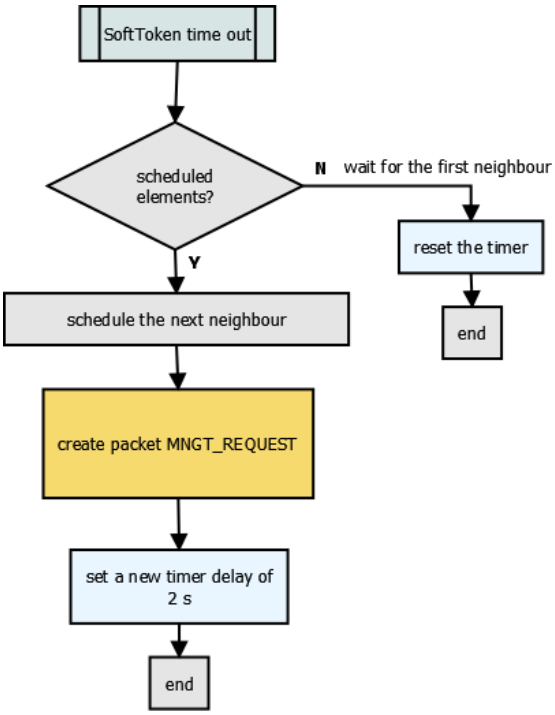


Figure 4.14: Flow diagram when a time out occurs

slave is added to the list of neighbours of the master, after scheduling itself, the master sets the slave as the current neighbour to send a request. It creates a *softtoken_mngt_request* which, this time, is transmitted by the wireless interface and again sets the timer to 2 seconds which is the hard-coded value for the retransmission timer. If after two seconds no *softtoken_mngt_response* has been received from the slave, a time out occurs which means that the token is assumed to be lost. This restarts the loop of scheduling the master, then the next neighbours if there is any. If the *softtoken_mngt_response* arrives before the time out occurs, the master updates the statistics for the transmitted and enqueued packets for the corresponding slave and again sets the timer to produce a time out immediately. The last possibility occurs when the *softtoken_mngt_response* arrives after the master has assumed that the token has been lost. This occurs at transient states in the beginning of the operation of SoftToken until SoftToken messages are synchronized. The late *softtoken_mngt_response* arrives producing a packet reception in the master. Master checks if the message is from the neighbour which is currently being requested (normal behaviour) or it corresponds to a late *softtoken_mgnt_response*. In this case, the master reschedules the neighbour whose response was lost as the next slave to be requested.

4.2.1.3 Scheduler

The configuration of the scheduler module determines the QoS in terms of bandwidth and TC, which is assigned per queue. This permits to guarantee different bandwidths for different types of traffic.

In the current release of SoftToken, the scheduler is configured manually in the master using the commands provided by the SoftToken Controller module(which are explained in detail in the next section). In the next section, we describe the functionality developed to permit some of the CARMEN primitives to modify dynamically the configuration of the scheduler after a resource allocation event: request, modify or release.
To better understand the behaviour of the scheduler, we provide its configuration parameters. The scheduler configuration is maintained in the master node. The information stored for each of the four queues of SoftToken are:

- the number of the queue
- the type of the resource allocation assigned to the queue, which can be: packets, bytes or time
- the value of the resource allocation assigned to the queue
- the traffic class assigned to the queue

The configuration of the scheduler is signalled by the master to the slaves using the *softtoken_mngt_request* messages. According to this configuration, a node releases for each queue the number of packets or bytes indicated by each *softtoken_mngt_request*.

4.2.2 User Space Implementation

The user space implementation is formed by the controller module. The controller is used to monitor, control and configure the kernel space of SoftToken from the user space by using ioctl calls. It is similar to the iwconfig tool for wireless cards. Some of the functionalities of the controller are:

- Setting the wireless interface which is going to be used by SoftToken
- Getting the interface used by SoftToken and the scheduler configuration (only in master mode)
- Adding and erasing slaves (only in master mode)
- Allocating a number of bytes or packets to a queue (only in master mode)
- Assigning the traffic class of a queue (only in master mode)
- Setting the time the scheduler waits for scheduling the next node (only in master mode)
- Displaying the MAC address, IP address, enqueued packets and bytes, and the mode of the node running SoftToken

4.3 Contributions

As it has been explained, the SoftToken release presented in the previous sections can only be configured by means of the SoftToken Controller in the user space. In order to integrate SoftToken into the CARMEN framework it is necessary to provide an interface to the Coordinated MAd, so that the AI primitives defined in section 3.5.5.1 coming from the IMF can properly interact with the SoftToken kernel module.
In this section we explain the extensions that have been developed for SoftToken and the Coordinated MAd to permit some of the technology-independent primitives to control and configure technology-specific SoftToken[2].

4.3.1 CARMEN Framework and SoftToken Integration

We will describe the integration of SoftToken inside the CARMEN framework in two steps. First, we present the modifications needed in the Coordinated MAd and SoftToken to enable the SCF to set the LGs and to enable the RtF to perform resource allocations. In order to achieve it, the SCF has to be able to run the SoftToken as the LGC, that is in master mode, by means of an *AI_Link_Set_Link_Group* request/response, and allow the RtF to perform one of the *AI_Link_Allocate_Resources*, *AI_Link_Modify_Resources* or *AI_Link_Release_Resources* request/response resource management calls. Second, we present the modifications needed to allow these primitives when the SCF launches SoftToken as a Link Agent (LA) that is in slave mode.

4.3.1.1 LG Setup and Resource Management in SoftToken

Figure 4.15 shows SoftToken within the CARMEN framework. The coloured boxes represent the parts which have been modified or added to integrate SoftToken. Black arrows represent the information flow for the initial setup of the LG using SoftToken. Magenta and turquoise arrows show the information flow for a resource management request/response in the master mode. Red and blue are for a resource management request/response in the slave node.

The first modification handles the set up of LGs which is performed by the *AI_Link_Set_Link_Group.request* primitive coming from the SCF in the Coordinated MAd. If the *NodeID* – which is a MAC address – is equal to the *LinkGroupID* it means that SoftToken has to be run in master mode because the *LinkGroupID* corresponds to the *NodeID* which has been selected by the SCF to be the LGC. Otherwise, SoftToken has to be run in slave node.
In order to do that, the handler in the Coordinated MAC executes the system calls to insert the SoftToken kernel module in master or slave mode as well as the interface to be used by SoftToken. In the master mode, a list with the MAC address of the nodes belonging to the LG

[2]Some of the functionalities have been done with the help of María Carolina Martínez Olmo, also an intern from Universidad Carlos III de Madrid at the T-Labs.

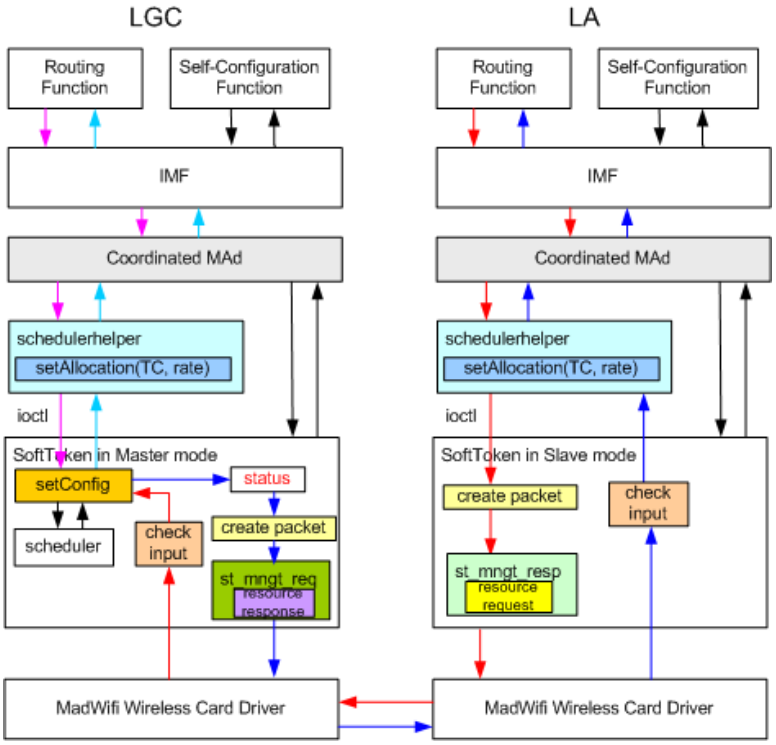


Figure 4.15: Integration of SoftToken in the CARMEN framework

is also provided by the SCF. If no error is reported, a *AI_Link_Set_Link_Group.response* with *success* status is sent. After this step, SoftToken is up and running. Next, we added support for the resource allocation functions. In order to handle the resource management – allocate, modify or release request/response – in the Coordinated MAd, two modules have been added:

- **Schedulerhelper** that contains the setAllocation function which receives from the Coordinated MAd the TC and the Rate of the allocation. This information is passed to SoftToken via a new ioctl which in turn calls the new setConfig module in the SoftToken kernel module and reports the status back triggering the corresponding response.
- **SetConfig** that contains the algorithm which dynamically adjusts the scheduler configuration in SoftToken taking into account the current status of the scheduler and the type of the incoming resource management request: allocation, modify or release. This module will be explained in detail later.

If a resource management – allocation, modify, release – request comes from the RtF, to a CARMEN node behaving as LA (i.e. running SoftToken in slave mode), we need to add to SoftToken the mechanism to signal the request to the master as well as to signal back the corresponding response. In order to do that, we need to modify the SoftToken packet formats to include the information of the resource management request or response and modify the corresponding SoftToken modules to handle them:

- In the schedulerhelper, we need to check whether we are in master or slave mode to be able to wait for the response to the resource management request. In order to do this, we add another ioctl to SoftToken which will report the status.
- Now, the ioctl called by the scheduler helper cannot call directly the setConfig module because in slave mode there is no such one. Instead of that, we set a flag which is read when the next *softtoken_mngt_response* is created, including the resource management request on it. To do that, we have modified the *softtoken_mngt_base* packet adding a field to indicate that the *softtoken_mngt_response* contains a resource management request.
- In the master node, the check input module verifies if the *softtoken_mngt_response* contains a resource management request in which case it calls the setConfig module which configures the scheduler accordingly and saves the status.
- Now the status needs to be sent back to the slave node. We use the field previously added to the *softtoken_mngt_base* to signal the status of the resource management request back to the slave in the next *softtoken_mngt_request* which is sent to the slave.
- In the slave node, the check input module verifies if the *softtoken_mngt_request* is carrying the response for the resource management request. In this case, it checks the status, which is reported back to the scheduler helper using the ioctl previously described.

4.3.1.2 Dynamic Scheduler Configuration

The Dynamic Scheduler uses the setConfig function and the static Scheduler configuration of SoftToken. The setConfig function is called whenever a resource management request is done in the LGC or in any LA. It is responsible for keeping track of the resource allocations, as well as for modifying and releasing them, taking into account:

- The current resource allocation.
- The TC of the resource request.
- The rate of the resource request.

The setConfig function implements an algorithm which provides a minimum bandwidth guarantee for each TC. It assumes that the total bandwidth that can be allocated among all TCs is fixed and corresponds to the saturation throughput for the link. Then, it allocates the bandwidth based on the water filling concept, trying to grant all the allocations when it is possible otherwise granting high priority allocations first.
Figure 4.16 shows the flow diagram for the algorithm. First, we compute how many TCs are in use. Next, we compute one of the 16 possible cases that can occur depending on the TC of the incoming allocation request. To cover these 16 cases, a water filling algorithm is implemented. In Figure 4.16 we depict the flow diagram for the water filling algorithm for the type of services 0 and 1. In the first case, as there is no allocation already done, the whole bandwidth

can be allocated. However, the *tos_case 1* corresponds to the case in which there is only one allocation for the TC 0. In this case, if the incoming request allocation is TC 0, it represents a modification of the current allocation which can be treated as the previous case. Otherwise, the incoming request allocation is of a TC with higher priority and in case there is no enough free bandwidth to grant the request, the existing lower priority allocation has to be decreased until a minimum value, so that the high priority request can be satisfied.

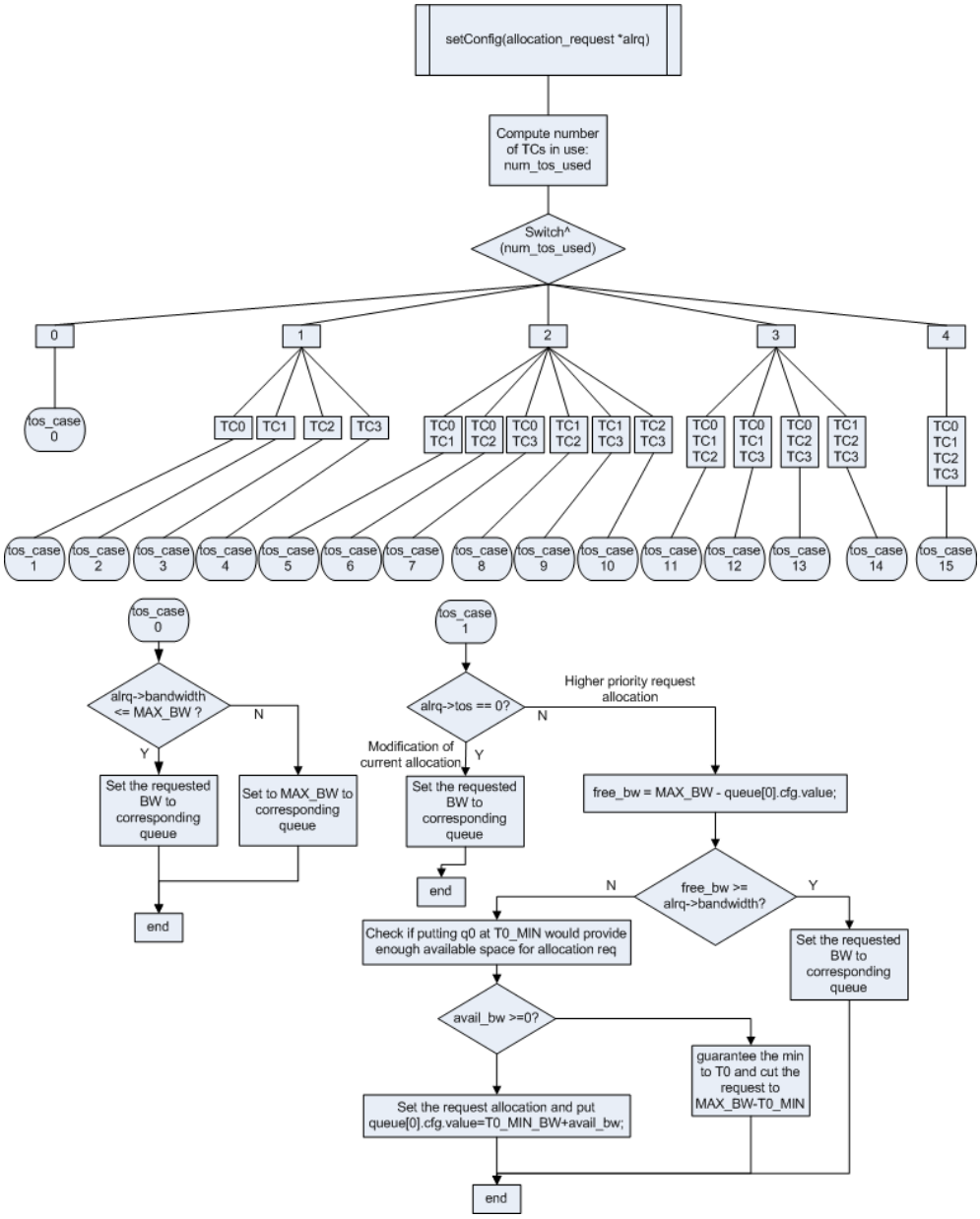


Figure 4.16: Flow diagram of the setConfig function

Chapter 5

Simple Analysis of SoftToken

5.1 Methodology

SoftToken works on top of the IEEE 802.11 protocol [7]. The analysis of the throughput of UDP communication over IEEE 802.11 [36] therefore, is needed to evaluate the performance when using SoftToken.
In this chapter, we study theoretically the throughput of UDP over IEEE 802.11 and the throughput when using SoftToken over IEEE 802.11a and we provide the results for different packet sizes.

Due to the distributed and random access nature of IEEE 802.11 it is not possible to provide a deterministic model for it. However, for simple scenarios, such as for two stations, we can provide some reasonably accurate results.
For this reason, we choose a two-node network topology to derive the theoretical analysis for both IEEE 802.11 and SoftToken. We study the case in which only one unidirectional UDP stream is being sent from one station to another. We call this scenario the baseline test which is intended to provide an upper bound on what it can be delivered in the worst case, firstly using IEEE 802.11a then using SoftToken. In the end, we present a use case for a specific type of UDP traffic – VoIP – which is of our interest due to its QoS requirements.

5.2 Throughput of IEEE 802.11a

IEEE 802.11a, operates in the 5 GHz band and uses OFDM modulation scheme. The data rates supported in IEEE 802.11a are 6, 9, 12, 18, 24, 36, 48 and 54 Mbps. We use IEEE 802.11a to avoid the interferences caused by the surrounding WLANs, most of them operating in with IEEE 802.11g in the 2.4 GHz band.

5.2.1 IEEE 802.11: Distributed Coordinated Function analysis

The DCF is the fundamental channel access method to share the medium between multiple stations and allows for automatic medium sharing through the use of:

- CSMA/CA
- Random backoff time following a busy medium condition
- Immediate Acknowledgement (ACK) frame where retransmission is scheduled by the sender if no ACK is received.

DCF defines two access modes:

- Two way handshake - basic access mode
- Four way handshake - Request To Send / Clear To Send (RTS/CTS) access mode

The time interval between frames is called the Interframe Space (IFS). A station determines that the medium is idle through the use of the carrier sense function for the interval specified. Five different IFSs are defined to provide priority levels for access to the wireless media. Figure 5.1 shows some of these relationships.

- SIFS: short interframe space
- PIFS: Point Coordinated Function (PCF) interframe space
- DIFS: DCF interframe space
- AIFS: arbitration interframe space (used by IEEE 802.11 for QoS support).
- EIFS: extended interframe space

The different IFSs are independent of the station data channel rate. The IFSs timings are defined as time gaps on the medium, and the IFSs timings except Arbitration IFS (AIFS) are fixed for each Physical Layer (PHY) – even in multirate-capable PHYs. Figure 5.1 shows the relationship between those IFSs.

5.2.2 The DCF two-way-handshake mechanism

From Figure 5.2 the transmission duration ($T_{TX_{2W}}$) of a data frame is:

$$T_{TX_{2W}} = DIFS + BT + T_{data} + SIFS + T_{ack} + 2 \cdot T_{prop} \tag{5.1}$$

The mean transmission duration ($T_{TX_{2Wmean}}$) can be calculated with:

$$T_{TX_{2Wmean}} = DIFS + BT_{mean} + T_{data} + SIFS + T_{ack} + 2 \cdot T_{prop} \tag{5.2}$$

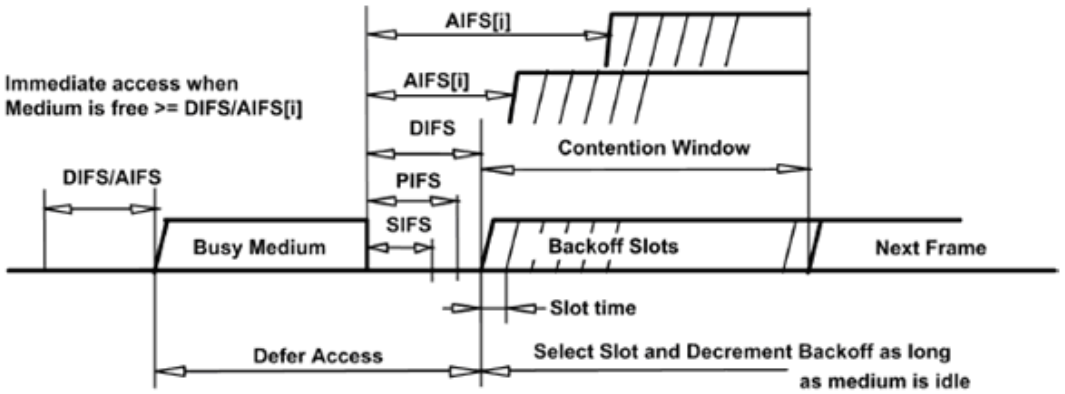


Figure 5.1: Some IFS relationships [7]

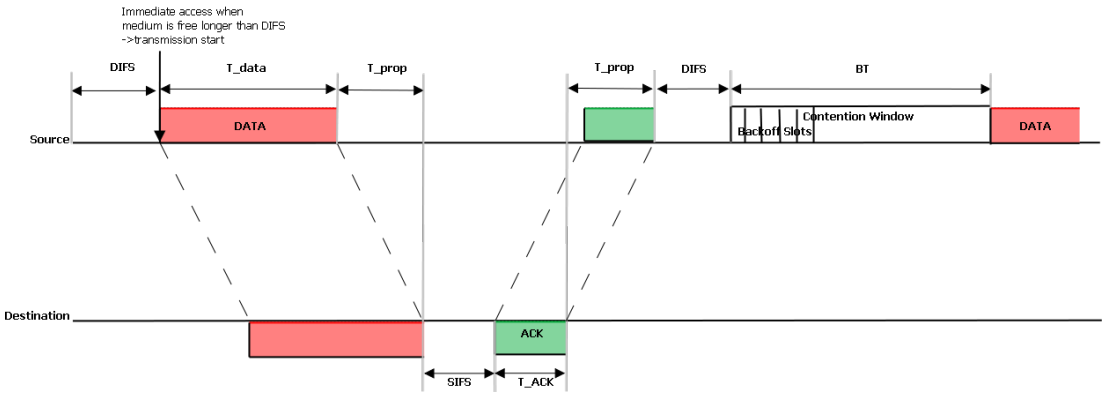


Figure 5.2: Frame transmission in the basic access mode

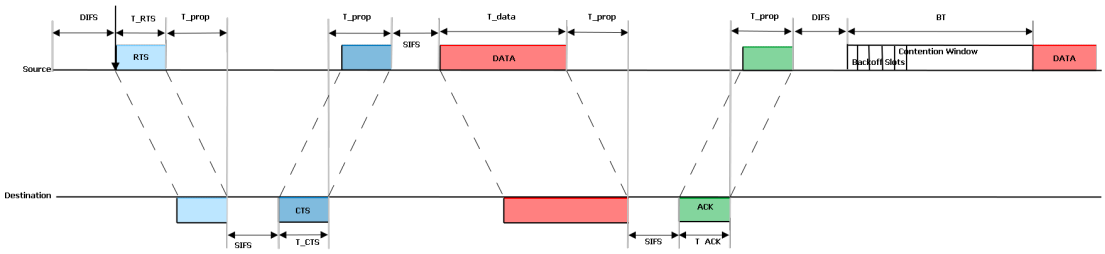


Figure 5.3: Frame transmission in the RTS/CTS access mode

5.2.3 The DCF four-way-handshake mechanism

From Figure 5.3 the transmission duration ($T_{TX_{4W}}$) of a data frame is:

$$T_{TX_{4W}} = DIFS + BT + T_{RTS} + SIFS + T_{CTS} + SIFS + T_{data} + SIFS + T_{ack} + 4 \cdot T_{prop} \quad (5.3)$$

The mean transmission duration ($T_{TX_{4W\,mean}}$) can be calculated with:

$$T_{TX_{4W\,mean}} = DIFS + BT_{mean} + T_{RTS} + SIFS + T_{CTS} + SIFS + T_{data} + SIFS + T_{ack} + 4 \cdot T_{prop} \quad (5.4)$$

The Backoff Time BT is calculated as:

$$BT = \sigma \cdot Random(0, CW - 1) \quad (5.5)$$

where σ is the slot time and CW is the Contention Window. After the first transmission attempt, CW is set to CW_{min} which is the minimum contention window. After each unsuccessful transmssion, CW is doubled, up to the maximum value CW_{max}. If we assume there are no other Stations STAs transmitting and thus no collisions during transmission, the value of CW_{min} will never increase. This assumption is valid if we say that we have only one WLAN transmitting STA and an optimal channel thus no collision occurs during transmission [36]. Then the mean value for the BT, BT_{mean} can be calculated with:

$$\begin{aligned} BT_{mean} &= \sigma \cdot Random(0, CW_{min} - 1) \\ &= \sigma \cdot 0.5 \cdot (0, CW_{min} - 1) \end{aligned} \quad (5.6)$$

5.2.4 Frame Transmission

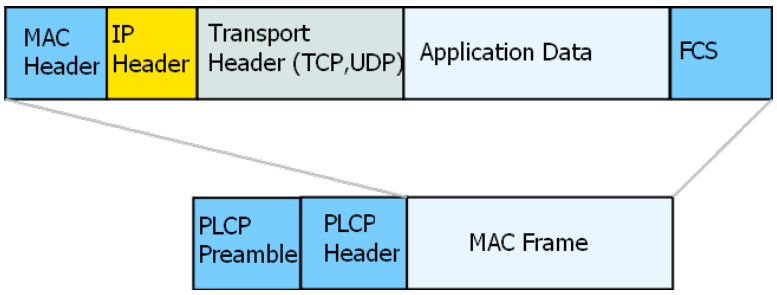


Figure 5.4: Data frame structure

A data frame consists of application data and some headers as shown in Figure 5.4. The size of the headers is:

- Wireless MAC Header + Frame Check Sequence (FCS) → 28 bytes

- IP Header: IPv4 → 20 bytes
- Transport Header: UDP → 8 bytes

Every frame (both data and control) is transmitted by means of Physical Layer Convergence Protocol (PLCP).

- PLCP Overhead → 192 bits or 120 bits depending if it is using the long or the short Preamble where:
  - PLCP Preamble → 144 bits or 72 bits. Used for synchronization.
  - PLCP Header → 48 bits. Contains information about transmission parameters for the MAC frame.

In 802.11a the PLCP overhead is transmitted at the default basic rate $R_{basic} = 6$ Mbps. However, the data rates R_{data} can be 6, 9, 12, 18, 24, 36, 48 and 54 Mbps.
The following formulas describe how to calculate the corresponding transmission times.

- Time to transmit a RTS packet (T_{RTS}):

$$T_{RTS} = \frac{PLCP_{length}}{R_{basic}} + \frac{RTS_{length}}{R_{data}} \tag{5.7}$$

- Time to transmit a CTS packet (T_{CTS}):

$$T_{CTS} = \frac{PLCP_{length}}{R_{basic}} + \frac{CTS_{length}}{R_{data}} \tag{5.8}$$

- Time to transmit an ACK packet (T_{ACK}):

$$T_{ACK} = \frac{PLCP_{length}}{R_{basic}} + \frac{ACK_{length}}{R_{data}} \tag{5.9}$$

- Time to transmit an UDP packet (T_{DATA}) with payload N bytes:

$$T_{DATA} = \frac{PLCP_{length}}{R_{basic}} + \frac{DATA_{length}}{R_{data}} \tag{5.10}$$

  where:

$$DATA_{length} = UDP_{header} + IP_{header} + WLAN_{MAC_{header}} + FCS + N \tag{5.11}$$

- Propagation time of a frame (T_{prop}) in seconds:

$$T_{prop} = \frac{d}{c} \tag{5.12}$$

  where d is the distance between the antennas in meters and c is the speed of light in m/s. All the transmission times are calculated in seconds.

5.2.5 Throughput Calculation

- Mean throughput (B_{2Wmean}) with two-way-handshaking mechanism:

$$\begin{aligned} B_{2Wmean} &= \frac{N}{T_{TX_{2Wmean}}} \\ &= \frac{N}{DIFS + BT_{mean} + T_{data} + SIFS + T_{ack} + 2 \cdot T_{prop}} \end{aligned} \tag{5.13}$$

- Mean throughput (B_{4Wmean}) with four-way-handshaking mechanism:

$$\begin{aligned} B_{4Wmean} &= \frac{N}{T_{TX_{4Wmean}}} \\ &= \frac{N}{DIFS + BT_{mean} + T_{RTS} + 3 \cdot SIFS + T_{CTS} + T_{data} + T_{ack} + 4 \cdot T_{prop}} \end{aligned} \tag{5.14}$$

Both equation 5.13 and 5.13 are the theoretical values for the maximum bandwidth that could be allocated to a STA assuming that:

- There is only UDP traffic.
- There are no collisions with other STAs.
- The MAC protocol is IEEE 802.11 with the DCF with fixed CW size.

Using the values in Table 5.1 and the previous formulas, we obtain the theoretical results for UDP throughput for IEEE 802.11 a in both access modes and for different values of data payload N. Table 5.2 and Table 5.3 show the numerical results for the 2-way handshake: $B_{2W_{mean}}$ and the 4-way handshake: $B_{4W_{mean}}$ respectively. For the sake of clarity, we only plot the results corresponding to the Table 5.2 in Figure 5.5 which provides an idea of the evolution of the UDP throughput for different link rates and UDP packet sizes.

If we compare Table 5.2 and Table 5.3 we can see that in all the cases, the 4-way-handshake achieves smaller throughput. This is due to the overhead caused by the RTS/CTS mechanism. Moreover, as equations 5.13 and 5.14 state, as the packet size increases, the throughput increases. If we compare the maximal throughput with its channel data rate, we can appreciate that it is for 6 Mbps that we obtain the best channel utilization. This is due to the fact that all the traffic – physical headers as well as UDP data – is being sent at the same basic data rate.

Parameter	**Value**
DIFS	50 μs
SIFS	10 μs
RTS_{length}	20 Bytes
CTS_{length}	14 Bytes
ACK_{length}	14 Bytes
$PLCP_{length}$ (short)	15 Bytes
UDP_{header}	8 Bytes
IP_{header}	20 Bytes
$WLAN_{MAC_{header}} + FCS_{length}$	28 Bytes
σ (SlotTime)	9 μs
CW_{min}	7
T_{prop}	0 μs
R_{basic}	6 Mbps
R_{data}	[6, 36, 54] Mbps

Table 5.1: 802.11a Parameters

N (Bytes)	$R_{data} = 6Mbps$	$R_{data} = 36Mbps$	$R_{data} = 54Mbps$
20	0.648	1.088	1.140
256	3.646	1.026	1.168
512	4.536	15.979	19.210
1024	5.166	22.134	28.339
1470	5.393	25.063	33.113

Table 5.2: UDP throughput for IEEE 802.11a in the DCF 2-way handshake mode

N (Bytes)	$R_{data} = 6Mbps$	$R_{data} = 36Mbps$	$R_{data} = 54Mbps$
20	0.454	0.746	0.779
256	3.070	7.670	8.521
512	4.062	12.646	14.720
1024	4.844	18.717	23.134
1470	5.145	21.908	27.987

Table 5.3: UDP throughput for IEEE 802.11a in the DCF 4-way handshake mode

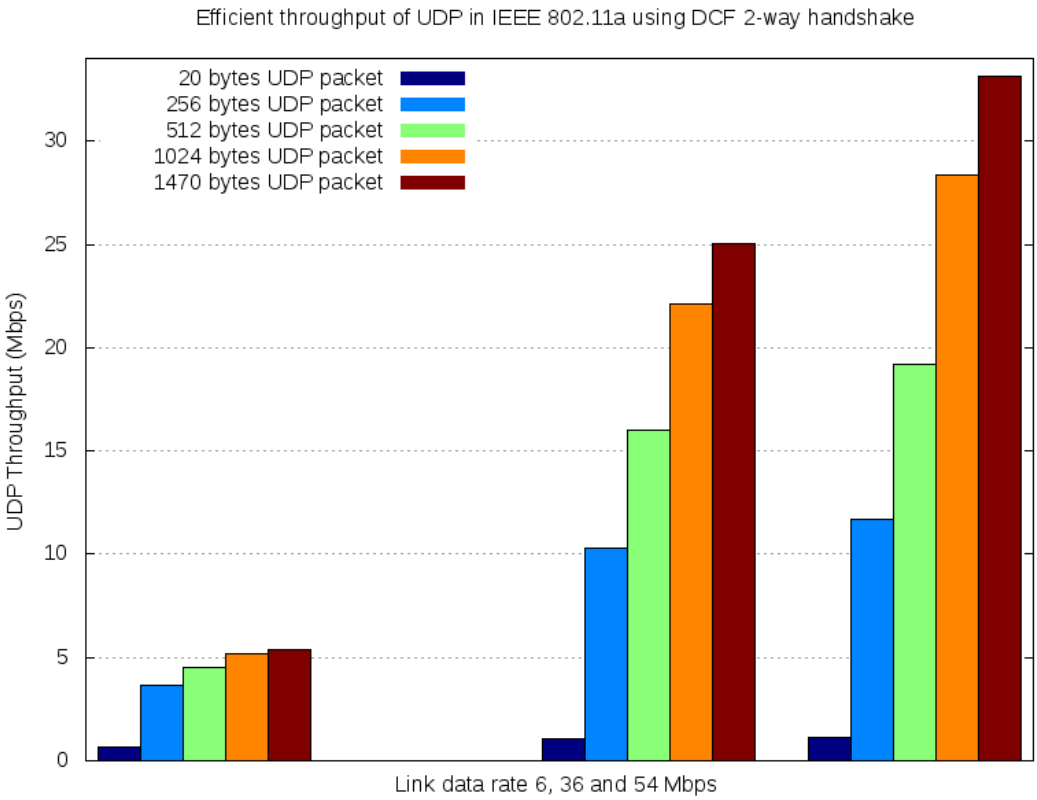


Figure 5.5: Comparison of UDP throughput for different packet sizes

5.3 Throughput of SoftToken over IEEE 802.11

In the previous section, we have obtained theoretically the throughput for IEEE 802.11a for a two-node topology. In this section, we use the same scenario to find theoretically the throughput for SoftToken and compare both. We use the same two-node topology as in Section 5.2as the baseline test to establish theoretically an upper bound on what SoftToken can deliver in its worst case configuration. Afterwards, we will apply this analysis in a specific case in which VoIP traffic is sent using UDP protocol [37].

5.3.1 Main logic of the baseline test

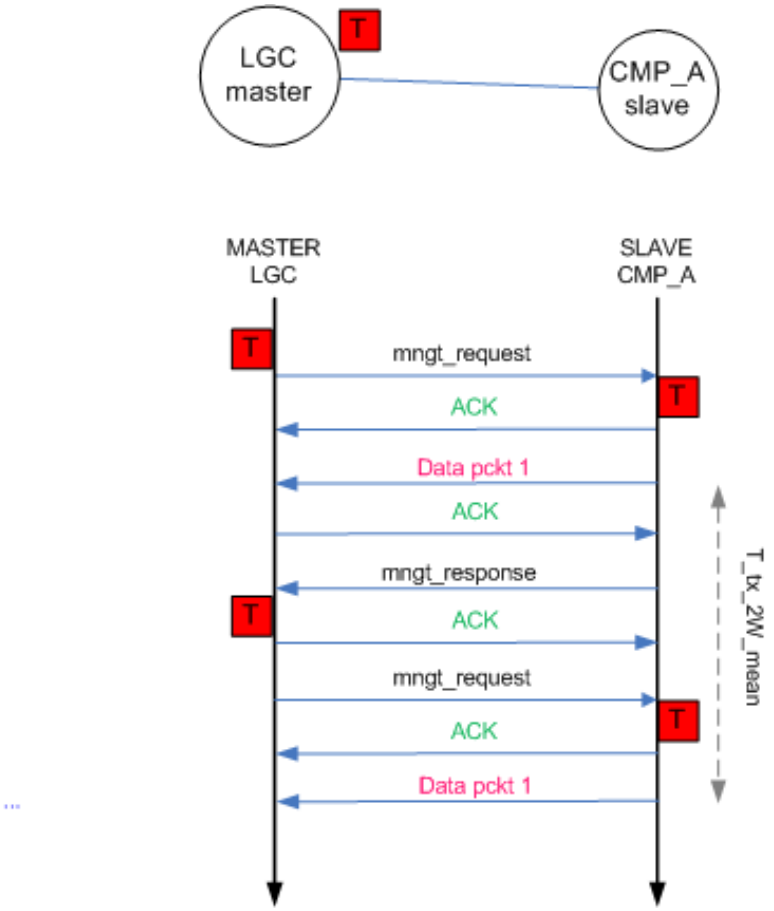


Figure 5.6: Baseline test topology

In the baseline test represented in Figure 5.6 a single CMP – CMP_A – working in slave mode transmits one single UDP data packet at a time to the LGC before passing the token back. This can be considered the worst-case configuration for SoftToken because for each data packet transmitted there are two SoftToken management packets: a SoftToken mngt_request and a SoftToken mngt_response.

Based on Figure 5.7 which describes the timing associated with the frame transmission in the baseline test shown in Figure 5.6, we calculate the efficient throughput of UDP using SoftToken. This constitutes an ideal model for SoftToken in which no additional processing times – such as SoftToken management packet creation and processing, packet enqueuing and packet dequeuing – have been taken into account.
In the baseline test only the slave has one type of UDP data to transmit to the master. The resource allocation is set in the master to one packet per queue. This allows the slave to transmit only one packet before giving the token back to the master. This behaviour is shown in Figure 5.6 which shows a MSC for the baseline test. As it can be seen, each IP datagram is acknowledged by an underlying IEEE 802.11 ACK frame. This can be seen as a violation of the token principle – the ACK after the first data packet is sent by the master which does not have the token – but it is not if we take into account that SoftToken only cares about the IP-based traffic and that it is working on the top of IEEE 802.11 making use of the reliable service that the latter provides.

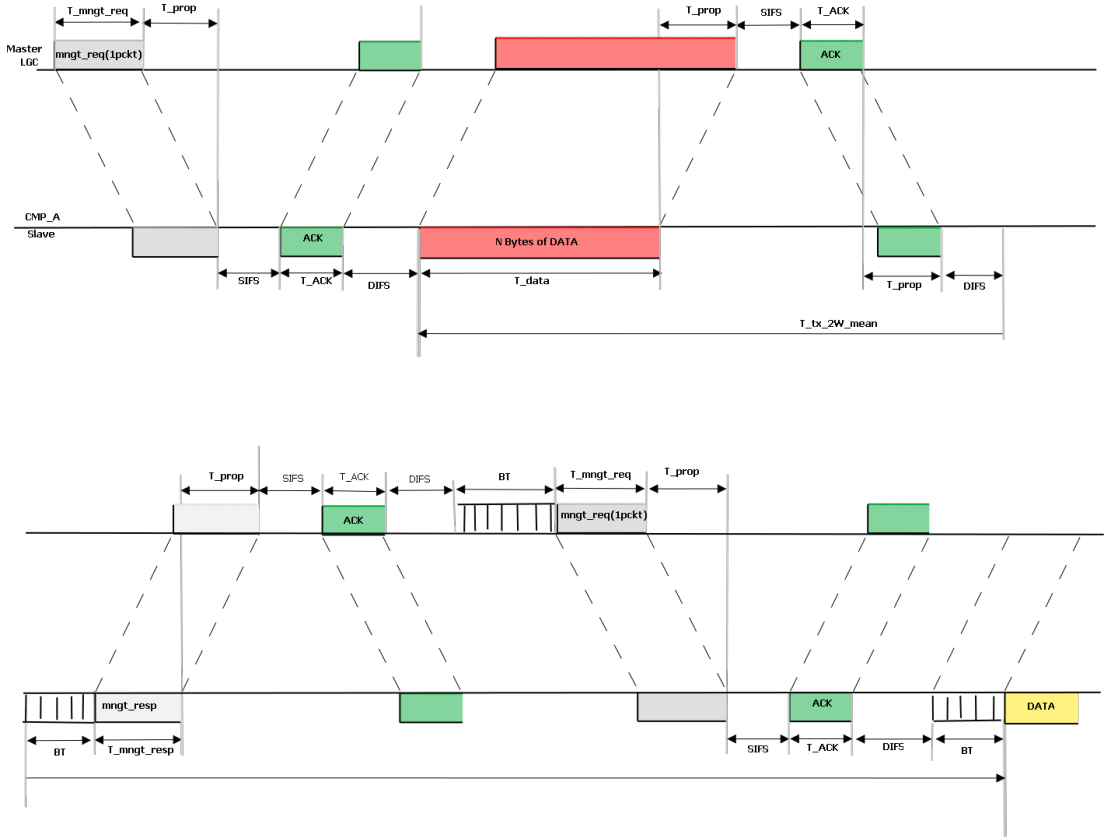


Figure 5.7: Frame transmission in the baseline test

5.3.2 Throughput Calculation

Applying the same analysis of Section 5.2 to the baseline test described in Figures 5.6 and 5.7, we find that the efficient throughput for SoftToken using the DCF two-way-handshake

mechanism (B_{2W}) is:

$$B_{2W} = \frac{N}{T_{TX_{2W\,mean}}} = \frac{N}{T_{data} + T_{mngt\_req} + T_{mngt\_resp} + 3 \cdot SIFS + 3 \cdot T_{ack} + 3 \cdot DIFS + 3 \cdot BT_{mean} + 4 \cdot T_{prop}} \tag{5.15}$$

where in this case the time to transmit a softtoken_management_request packet T_{mngt_req} with payload $MNGT\_REQ$ in bytes is:

$$T_{mngt\_req} = \frac{PLCP_{length}}{R_{basic}} + \frac{mngt\_req_{length}}{R_{data}} \tag{5.16}$$

where:

$$mngt\_req_{length} = UDP_{header} + IP_{header} + WLAN_{MAC_{header}} + FCS + MNGT\_REQ \tag{5.17}$$

and equivalently the time to transmit a softtoken_management_response packet T_{mngt_resp} with payload $MNGT\_RESP$ in bytes is:

$$T_{mngt\_resp} = \frac{PLCP_{length}}{R_{basic}} + \frac{mngt\_resp_{length}}{R_{data}} \tag{5.18}$$

where:

$$mngt\_resp_{length} = UDP_{header} + IP_{header} + WLAN_{MAC_{header}} + FCS + MNGT\_RESP \tag{5.19}$$

Using the values in Table 5.4 and the previous equations, we obtain the theoretical results for UDP throughput using SoftToken over IEEE 802.11a. These results are shown in Table 5.5 and plotted in Figure 5.8. We also compute the contribution of the terms in the $T_{TX_{2W\,mean}}$ to quantify the overhead introduced by SoftToken for 20 B and 1470 B UDP datagrams. This is shown in Tables 5.6 and 5.7 respectively.

Parameter	Value
DIFS	50 μs
SIFS	10 μs
ACK_{length}	14 Bytes
$PLCP_{length}$ (short)	15 Bytes
N	20 Bytes
UDP_{header}	8 Bytes
IP_{header}	20 Bytes
$WLAN_{MAC_{header}} + FCS_{length}$	28 Bytes
σ (SlotTime)	9 μs
CW_{min}	7
T_{prop}	0 μs
R_{basic}	6 Mbps
R_{data}	[6, 36, 54] Mbps
R_{VoIP}	8 Kbps
MNGT_REQ payload	50 bytes
MNGT_RESP payload	102 bytes

Table 5.4: Parameters

N (Bytes)	R_{data} = 6 Mbps	R_{data} = 36 Mbps	R_{data} = 54 Mbps
20	0.179	0.343	0.366
256	1.699	3.951	4.334
512	2.648	7.121	8.024
1024	3.675	11.890	13.972
1470	4.164	14.922	18.026

Table 5.5: Efficient throughput of UDP in Mbps using SoftToken over IEEE 802.11a

Component	Time (μs)	%
T_{data}	31.25	7.1
T_{mngt_req}	35.703	8.2
T_{mngt_resp}	43.41	10
$3 \cdot SIFS$	30	6.8
$3 \cdot T_{ack}$	66.222	15.1
$3 \cdot DIFS$	150	34.3
$3 \cdot BT_{mean}$	81	18.5
Total	437.58	100

Table 5.6: Component contribution in the transmission time $T_{TX_{2Wmean}}$ for 20 bytes UDP datagrams at 54 Mbps

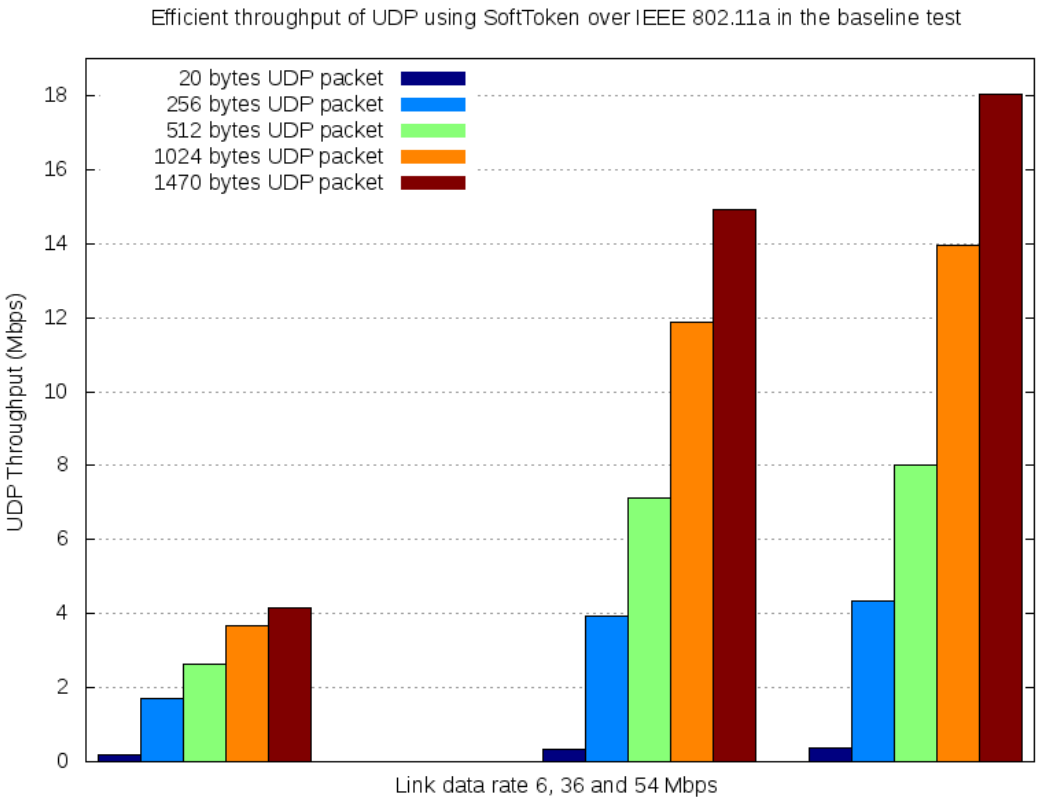


Figure 5.8: Efficient throughput of UDP using SoftToken over IEEE 802.11a 5.5

Component	**Time** (μs)	**%**
T_{data}	246.074	37.7
T_{mngt_req}	35.703	5.5
T_{mngt_resp}	43.41	6.6
$3 \cdot SIFS$	30	4.6
$3 \cdot T_{ack}$	66.222	10.2
$3 \cdot DIFS$	150	23
$3 \cdot BT_{mean}$	81	12.4
Total	652.41	100

Table 5.7: Component contribution in the transmission time $T_{TX_{2Wmean}}$ for 1470 bytes UDP datagrams at 54 Mbps

5.3.3 Use Case: VoIP in SoftToken

As it has been explained, one of the goals of SoftToken is to guarantee a certain QoS depending on the type of the traffic. A typical QoS-sensitive application is Voice over IP (VoIP). A VoIP call can be emulated by two unidirectional streams. The characteristics of those streams depend on the codec. Taking the G.729 voice codec specifications, VoIP data is packetized into 20 bytes UDP payload packets every 20 ms which gives a bit rate of 8 Kbps for each stream. Other characteristics of VoIP connections are that the maximum end-to-end path delay has to be below 150 ms, the packet loss ratio must be less than 1 % to avoid audible errors and the jitter must be lower than 1 ms [38].
We are going to use the baseline test described in section 5.3.1 to calculate the number of unidirectional aggregated VoIP connections that can be supported using SoftToken for different channel rates. The number of unidirectional aggregated VoIP connections $\#VoIP_{con}$ is given by:

$$\#VoIP_{con} = \left\lfloor \frac{B_{2W}}{R_{VoIP}} \right\rfloor \tag{5.20}$$

Using the values for B_{2W} obtained in Table 5.2 for 802.11 and in Table 5.5 for SoftToken, for a UDP payload of 20 bytes we can compute $\#VoIP_{con}$ for the different channel data rates. The results are shown in Table 5.8.

R_{data} **(Mbps)**	$\#VoIP_{con}$ **IEEE 802.11**	$\#VoIP_{con}$ **SoftToken**
6	81	22
36	136	42
54	142	45

Table 5.8: Number of one-way VoIP connections

From Table 5.8 we see that $\#VoIP_{con}$ in the baseline test is lower for SoftToken when compared to IEEE 802.11. The difference is explained by the overhead introduced by the token passing mechanism in SoftToken. Note also that this scenario is a non-problematic scenario for IEEE 802.11 as there is no channel contention among competing nodes.

Chapter 6

Validation and Performance Tests

In order to validate the theoretical analysis presented in Section 5 we have run several tests varying:

- Transport-layer protocol: UDP or TCP
- Direction of the traffic streams: unidirectional or bidirectional
- Number of traffic classes: pure traffic classes or mixed traffic classes

The hardware we have used to emulate the CARMEN mesh nodes in the tests consists of laptops with Intel Pentium M processor at 1.73 GHz and 1 GB of RAM memory equipped with Lancom Air Lancer MC-54ag wireless PCMCIA cards with Atheros chipsets. Laptops run Ubuntu 9.10 with Linux kernel 2.6.31 and madwifi-0.9.4-4086 [39] is used as the wireless driver. For all the tests we used Iperf [40] traffic generator to create different UDP and TCP streams, and Tcpdump [41] and Wireshark [42] protocol analyzers to intercept and display the packets transmitted by the wireless interfaces.

6.1 UDP Tests

6.1.1 Two-node Baseline Test

In this section, we first set up a testbed equivalent to the baseline test explained in section 5.3.1 to evaluate the performance in a real scenario. Finally, we compare the performance of SoftToken with IEEE 802.11a.
The testbed corresponding to the baseline test described in the previous section was set up using two laptops: the LGC laptop running in the softtoken master mode and the other one

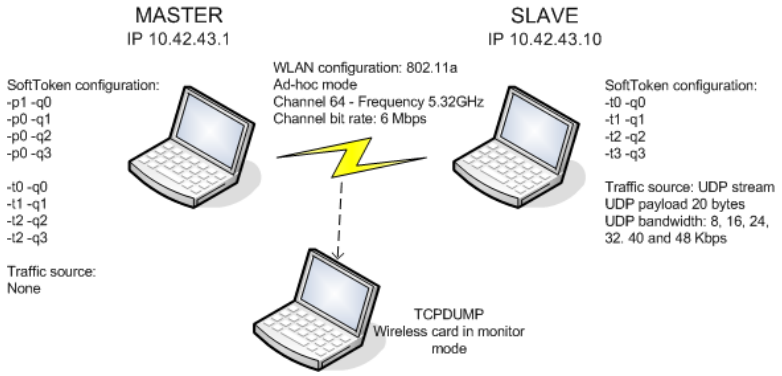


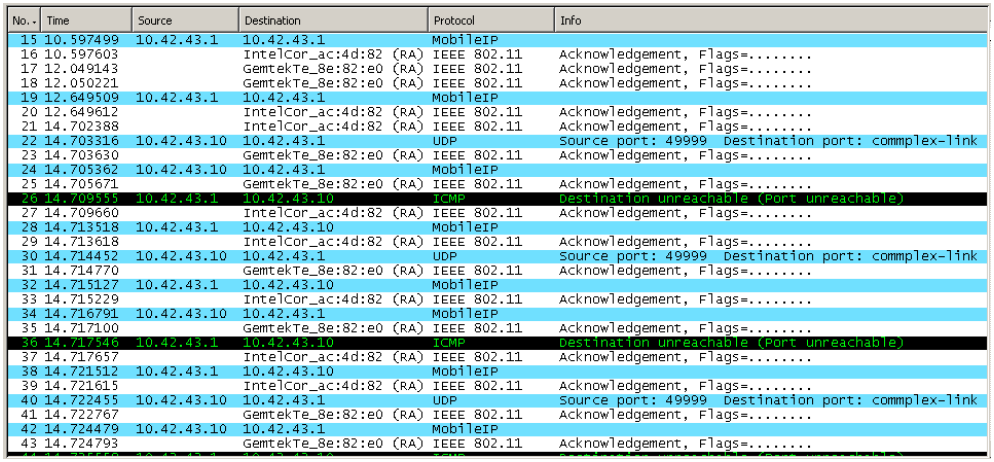
Figure 6.1: Scenario for the baseline testbed

in the slave mode. This is shown in Figure 6.1. To experimentally find the upper bound for the baseline test, we start with a single unidirectional VoIP connection as the traffic source. We then compare the results for the effective throughput with the theoretical ones obtained in section 5.3 and determine if the upper bound has been reached. If it has not been reached, we emulate that we add another unidirectional VoIP connection and we again compare the results till the upper bound is reached. We set up the Iperf application in each node to emulate the VoIP traffic and obtain the statistics. The LGC is configured as the UDP server that detects packet loss, performs jitter calculation and relative transit time (servers receive time - clients send time). The mesh node is configured as the UDP client. In order to emulate the unidirectional VoIP traffic each unidirectional voice stream has a UDP bandwidth equal to 8 Kbps, the UDP payload per packet is set to 20 bytes and we send 2000 bytes of data which means 100 packets for each unidirectional connection. To be able to monitor the frame exchange between the master and the slave node, we capture all the wireless traffic in a third laptop with its wireless card in monitor mode using Tcpdump.

6.1.1.1 Frame Exchange Analysis for SoftToken-0.1.0

Figure 6.2 shows a screenshot of the captured frames in the monitor laptop for one unidirectional VoIP connection. The IP address 10.42.43.1 corresponds to the LGC and the 10.42.43.10 corresponds to the slave node, the SoftToken request and response frames are recognized as MobileIP protocol and the UDP frames correspond to the VoIP emulated data.
Figure 6.3 shows a frame sequence diagram *translating* the capture shown in Figure 6.2 into meaningful messages. Frames which do not follow the normal behaviour of SoftToken are depicted in red in Figure 6.3. At the beginning of the operation there is a transient time where time-outs and retransmissions occur until the nodes are synchronized. During the initial tests, we noticed the problem of *ICMP destination unreachable (port unreachable)* messages occuring during transient states. These ICMP messages were a result of a bug in SoftToken affecting its performance, therefore we needed to debug the source code to solve the problem.
Taking a look at the source code we found that in the kernel space implementation, after checking an incoming SoftToken management packet, instead of discarding the packet, SoftToken

allowed it to pass. This is what generated the ICMP error messages. Allowing SoftToken to discard theses packets solved the bug of the ICMP error messages. In the next section, we analyse the frame exchange for the debugged version of SoftToken.



No.	Time	Source	Destination	Protocol	Info
15	10.597499	10.42.43.1	10.42.43.1	MobileIP	
16	10.597603		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flags=........
17	12.049143		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flags=........
18	12.050221		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flags=........
19	12.649509	10.42.43.1	10.42.43.1	MobileIP	
20	12.649612		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flags=........
21	14.702388		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flags=........
22	14.703316	10.42.43.10	10.42.43.1	UDP	Source port: 49999 Destination port: commplex-link
23	14.703630		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flags=........
24	14.705362	10.42.43.10	10.42.43.1	MobileIP	
25	14.705671		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flags=........
26	14.709513	10.42.43.1	10.42.43.10	ICMP	Destination unreachable (Port unreachable)
27	14.709660		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flags=........
28	14.713518	10.42.43.1	10.42.43.10	MobileIP	
29	14.713618		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flags=........
30	14.714452	10.42.43.10	10.42.43.1	UDP	Source port: 49999 Destination port: commplex-link
31	14.714770		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flags=........
32	14.715127	10.42.43.1	10.42.43.10	MobileIP	
33	14.715229		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flags=........
34	14.716791	10.42.43.10	10.42.43.1	MobileIP	
35	14.717100		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flags=........
36	14.717546	10.42.43.1	10.42.43.10	ICMP	Destination unreachable (Port unreachable)
37	14.717657		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flags=........
38	14.721512	10.42.43.1	10.42.43.10	MobileIP	
39	14.721615		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flags=........
40	14.722455	10.42.43.10	10.42.43.1	UDP	Source port: 49999 Destination port: commplex-link
41	14.722767		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flags=........
42	14.724479	10.42.43.10	10.42.43.1	MobileIP	
43	14.724793		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flags=........

Figure 6.2: Frame capture for the baseline test showing the ICMP error messages

6.1.1.2 Frame Exchange Analysis for the Debugged Version of SoftToken

In this section we show another capture of the frame exchange for the baseline test but for the debugged version of the SoftToken. This is shown in Figure 6.4. Again we provide a *translation* of SoftToken messages in Figure 6.5.

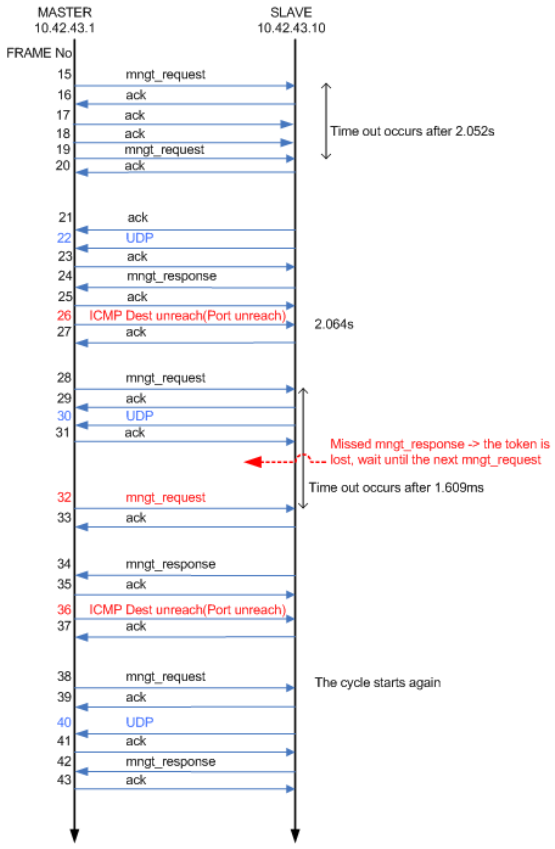


Figure 6.3: Message sequence chart for the baseline test capture of Figure 6.2

No.	Time	Source	Destination	Protocol	Info
77	14.764813		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flag
78	14.769500	10.42.43.1	10.42.43.10	MobileIP	
79	14.769603		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flag
80	14.770442	10.42.43.10	10.42.43.1	UDP	Source port: 49999 D
81	14.770754		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flag
82	14.772485	10.42.43.10	10.42.43.1	MobileIP	
83	14.772799		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flag
84	14.777499	10.42.43.1	10.42.43.10	MobileIP	
85	14.777602		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flag
86	14.778441	10.42.43.10	10.42.43.1	UDP	Source port: 49999 D
87	14.778754		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flag
88	14.780310	10.42.43.10	10.42.43.1	MobileIP	
89	14.780620		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flag
90	14.785498	10.42.43.1	10.42.43.10	MobileIP	
91	14.785601		IntelCor_ac:4d:82 (RA)	IEEE 802.11	Acknowledgement, Flag
92	14.786445	10.42.43.10	10.42.43.1	UDP	Source port: 49999 D
93	14.786753		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flag
94	14.788404	10.42.43.10	10.42.43.1	MobileIP	
95	14.788717		GemtekTe_8e:82:e0 (RA)	IEEE 802.11	Acknowledgement, Flag

Figure 6.4: Frame capture for the baseline test showing a correct behaviour

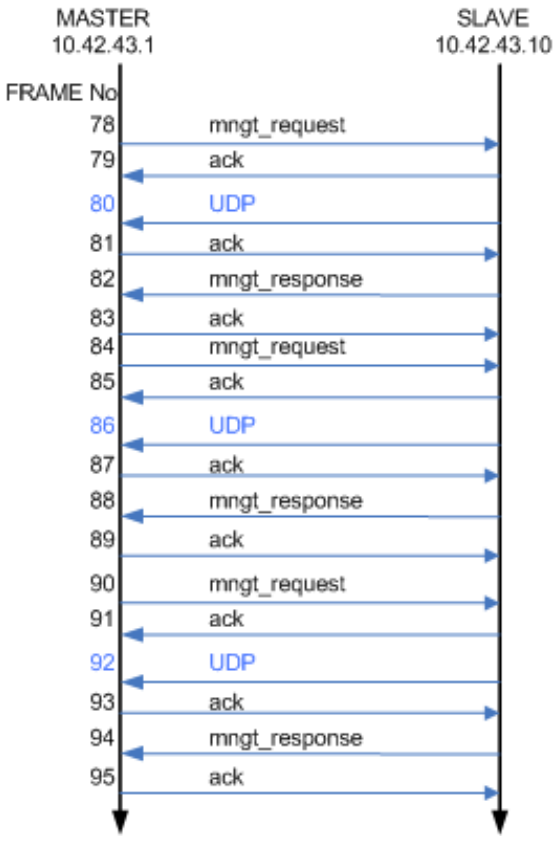


Figure 6.5: Message sequence chart for the baseline test capture of Figure 6.4

6.1.1.3 Evaluation for Undirectional VoIP Connections

The VoIP emulated traffic had a packetization interval of 20 ms. However, if we observe the time elapsed between the two successful data transmissions in Figure 6.4, as between Frame No. 80 and Frame No. 86, we find that it is in average around 8 ms. This means, as we already know, that VoIP packets are enqueued by SoftToken until they can be transmitted respecting the token principle. This queue time added to the time the token takes to return to the same node introduces latency. This latency is a critical parameter for QoS sensitive applications and thus it will be necessary to analyze it carefully. In Table 6.1 we summarize the results obtained with Iperf for different number of unidirectional VoIP connections and different number of requested packets per queue in the SoftToken configuration. As the number of VoIP connections increase, we will need to increase the number of requested packets per queue to be able to meet the bandwidth requirements.

#VoIP conn.	**Generated BW (Kbps)**	**Delivered BW (Kbps)**	**Jitter (ms)**	**Number pckt/queue**
1	8	8	13.585	1
2	16	16	9.078	1
3	24	19.1	7.855	1
3	24	24	8.416	2
4	32	32	5.700	2
5	40	37.6	5.143	2
5	40	39.4	5.954	3
6	48	46.4	4.909	3

Table 6.1: Experimental results for the baseline test using SoftToken

As we see from Table 6.1 the maximum bandwidth which can be delivered by SoftToken for the baseline test depends on the configuration of the number of allowed packets to send per queue. As we increase it, more packets can be sent in a single request-transmit-response cycle which means that we increase the effective bandwidth.

6.1.2 Throughput Comparison with SoftTDMAC in Two-node Tests

In this section, we present the results obtained for the UDP unidirectional saturation throughput tests for different channel rates and different configurations of SoftTDMAC[1], SoftToken and 802.11a. These tests were run in the same location, using the same radio channel and alternatively in time so that they were affected by the same environmental interferences and thus providing comparable results.
We find very useful to run the same tests for SoftTDMAC and SoftToken to compare the performance of the two approaches proposed by the TLabs for the Coordinated MAC in the CARMEN project.
The three configurations for the resource allocation used in SoftTDMAC are:

[1] SoftTDMAC tests were done by Efe Cullu who is a student at the Technische Universität Berlin researching on SoftTDMAC for the TLabs.

- Configuration 20%M-70%S. The frame slots are allocated as: 10% control, 20% master, 70% slave
- Configuration 45%M-45%S. The frame slots are allocated as: 10% control, 45% master, 45% slave
- Configuration 70%M-20%S. The frame slots are allocated as: 10% control, 70% master, 20% slave

The three configurations for the resource allocation used in SoftToken are:

- 8 packets or 11,760 bytes for queue 0, no packets for the rest
- 16 packets or 23,520 bytes for queue 0, no packets for the rest
- 32 packets or 47,040 bytes for queue 0, no packets for the rest

We use only one traffic class with ToS=0 which is sent in SoftToken through queue 0 and there is only one UDP stream sent from the Slave to the Master.
Each test was run for 100 seconds, obtaining every 10 seconds the Iperf statistics for the throughput, the jitter and the packet loss. For the results obtained for SoftToken and 802.11a we compute the average and the standard deviation for these values. Figure 6.6 shows the configuration of the testbed for SoftToken in this case.

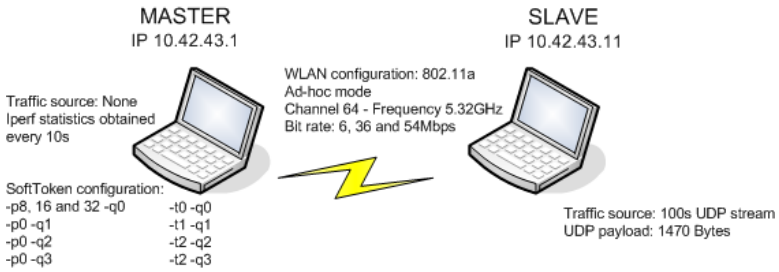


Figure 6.6: Configuration for SoftToken the unidirectional UDP test

The results for the UDP saturation throughtput for SoftTDMAC are shown in Table 6.2 and the ones for SoftToken and 802.11a as well as its error are shown in Table 6.3. For SoftTDMAC and SoftToken we highlight the best results in throughput. In order to compare them, they are plotted together in Figure 6.7.

As it was expected taking into account the theoretical analysis of Section 5.3, the channel data rate for which we obtain the best performance of SoftTDMAC and SoftToken compared to 802.11 is 6 Mbps. For 36 Mbps and 54 Mbps the difference in the throughput between the Coordinated MACs and 802.11 is between 10 and 15 Mbps which means that in this scenario it makes no sense the use of either of the Coordinated MACs instead of the standard 802.11a.
However, the results show that SoftToken is able to get 2-3 times more throughput than SoftTDMAC. The results of [33] for SoftTDMAC explain the loss of efficiency due to control message

Link data rate (Mbps)	SoftTDMAC thrp. for 20%M-70%S (Mbps)	SoftTDMAC thrp. for 45%M-45%S (Mbps)	SoftTDMAC thrp. for 70%M-20%S) (Mbps)
6	2.89	1.75	0.588
36	8.91	8.90	4.69
54	9.84	9.79	6.27

Table 6.2: UDP unidirectional saturation throughput for SoftTDMAC

Link data rate (Mbps)	SoftToken thrp. for 8 pckt/q (Mbps)	SoftToken thrp. for 16 pckt/q (Mbps)	SoftToken thrp. for 32 pckt/q (Mbps)	802.11a thrp. (Mbps)
6	3.309 ± 0.177	4.242 ± 0.251	4.295 ± 0.298	5.260 ± 0.032
36	10.273 ± 1.184	14.320 ± 1.249	15.790 ± 0.735	24.690 ± 0.110
54	9.808 ± 1.517	13.580 ± 1.736	18.010 ± 2.403	27.160 ± 0.341

Table 6.3: UDP unidirectional saturation throughput and its error for SoftToken and IEEE 802.11a

exchange and memory management. Nevertheless, as Table 6.4 shows, SoftTDMAC provides better results in terms of jitter due to the TDMA based scheduling.

As there is only traffic from the slave to the master it is logical that the configuration of SoftTDMAC which achieves better throughput is the one that assigns a bigger slot to the slave: 20%M-70%S. For SoftToken we can see the effect of the number of packets assigned to queue 0. As it is also logical, the best UDP throughput is achieved for the biggest number: 32 packets to queue 0, as there is less overhead in terms of SoftToken management packets.

6.1.2.1 UDP average jitter for SoftTDMAC, SoftToken and 802.11a

In Table 6.4 and Table 6.5 we show the results for the average jitter obtained for the UDP unidirectional test for SoftTDMAC, SoftToken and IEEE 802.11a respectively, where, again the best results for SoftTDMAC and SoftToken are highlighted. These values are plotted together in Figure 6.8.

Link data rate (Mbps)	SoftTDMAC jitter for 20%M-70%S (ms)	SoftTDMAC jitter for 45%M-45%S (ms)	SoftTDMAC jitter for 70%M-20%S) (ms)
6	3.120	6.334	8.243
36	0.104	0.115	2.154
54	0.058	0.055	1.204

Table 6.4: UDP unidirectional average jitter for SoftTDMAC

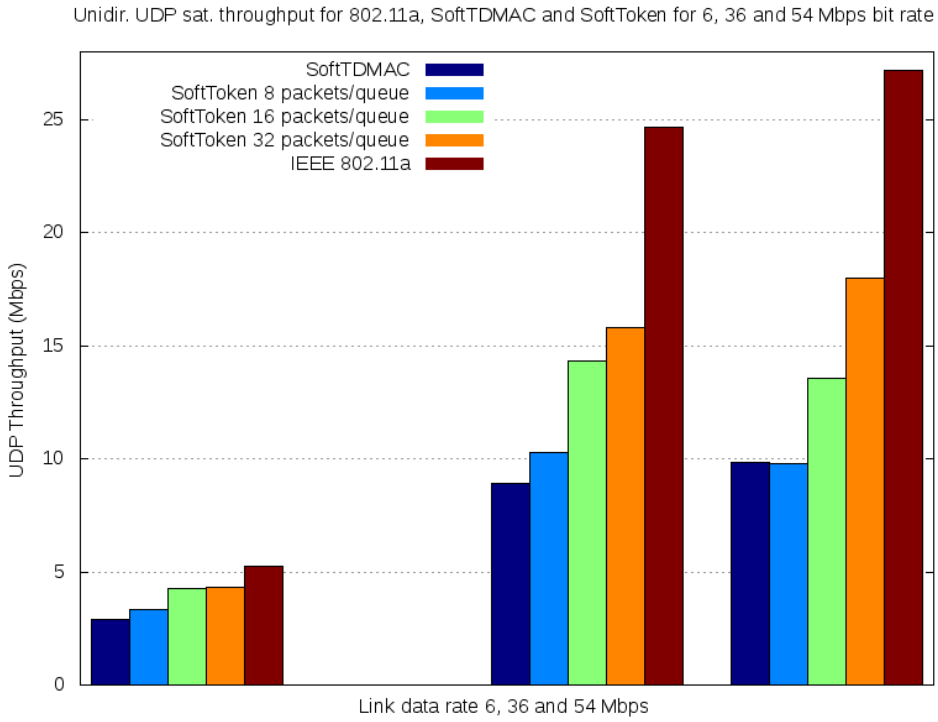


Figure 6.7: UDP unidirectional saturation throughput for SoftTDMAC, SoftToken and IEEE 802.11a

Link data rate (Mbps)	**SoftToken thrp. for 8 pckt/q (ms)**	**SoftToken thrp. for 16 pckt/q (ms)**	**SoftToken thrp. for 32 pckt/q (ms)**	**802.11a thrp. (Mbps)**
6	4.602	4.603	3.997	4.356
36	1.252	1.0037	1.3681	3.047
54	6.614	0.997	1.002	0.5446

Table 6.5: UDP unidirectional average jitter for SoftToken and IEEE 802.11a

In terms of jitter, the best values are obtained for SoftTDMAC as it is based on TDMA which provides strict timing. For SoftToken, except the atypical value for 8 packets for queue 0 at 54 Mbps, we can see that resource allocation parameter does not have a significant effect on the jitter. In general, we can see that the jitter decreases as the channel data rate increases.

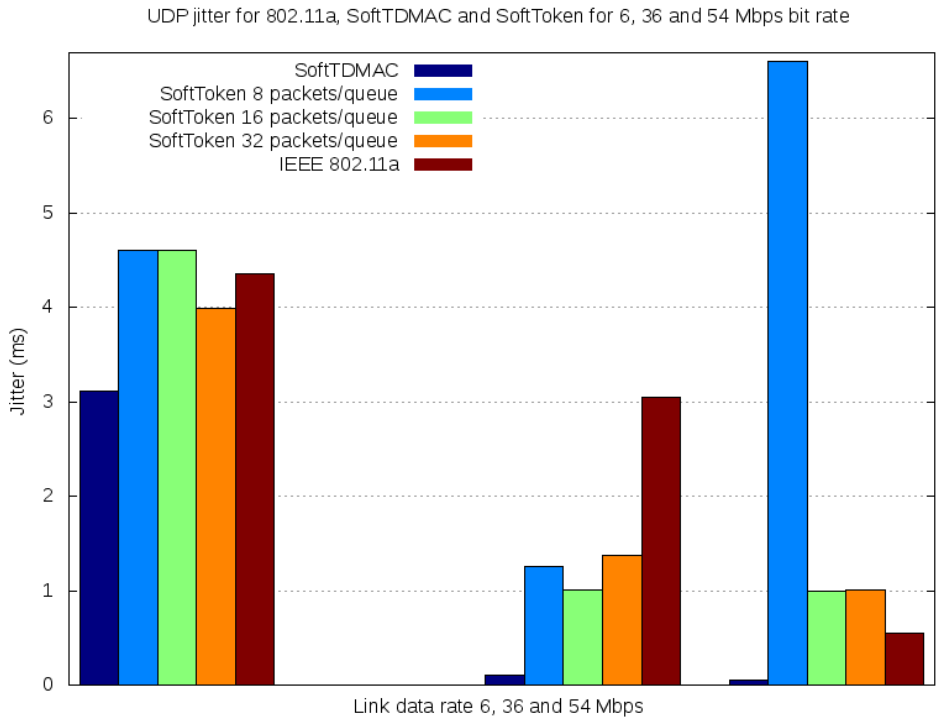


Figure 6.8: UDP unidirectional average jitter for SoftTDMAC, SoftToken and IEEE 802.11a

6.1.2.2 UDP average packet loss for SoftToken and 802.11a

In Table 6.6 we show the results for the average packet loss obtained for the UDP unidirectional test for SoftToken and IEEE 802.11a. These values are plotted together in Figure 6.9. As it

Link data rate (Mbps)	**SoftToken pckt.loss loss 8 pckt/q (%)**	**SoftToken pckt. loss 16 pckt/q (%)**	**SoftToken pckt. loss 32 pckt/q (%)**	**802.11a pckt. loss (%)**
6	0.035	0.075	0.712	0.099
36	0.066	0.036	0.615	0.767
54	0.037	0.051	0.165	8.710

Table 6.6: UDP unidirectional average packet loss for SoftToken and IEEE 802.11a

can be seen, the packet loss for SoftToken is below 1% which is an acceptable value for VoIP, however, for 802.11a as the transmit rate increases, the packet loss also increases, reaching a peak of 8% of packet losses at 54 Mbps.

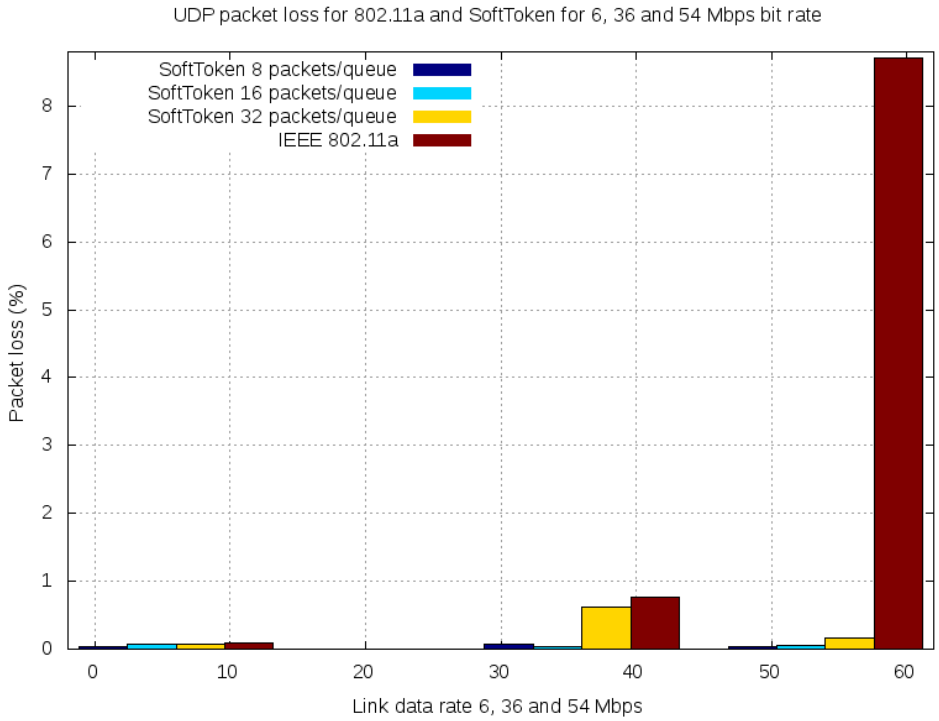


Figure 6.9: UDP unidirectional average packet loss for SoftToken and IEEE 802.11a

6.1.3 Two-node Bidirectional Saturation Throughput

6.1.3.1 Saturation Throughput for Different Resource Allocations at 54 Mbps

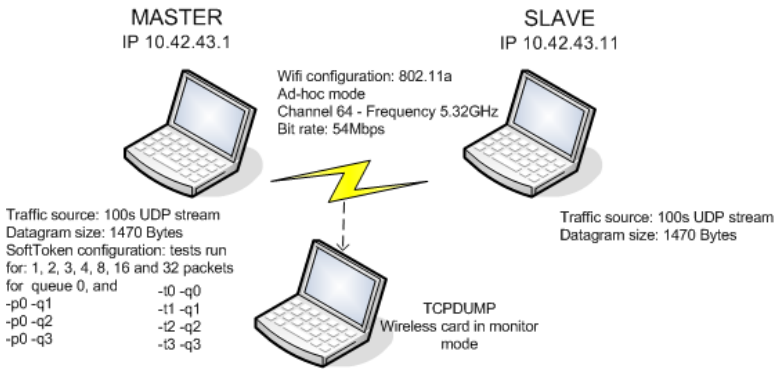


Figure 6.10: Configuration for the bidirectional UDP test

In this section, we obtain the UDP saturation throughput in a bidirectional test for different resource allocations of SoftToken and for a channel rate of 54 Mbps. Once we have these results, we choose the configurations of SoftToken which provide the best results to obtain the bidirectional UDP throughput also for 6 and 36 Mbps channel data rates.

To find the UDP saturation throughput using SoftToken with bidirectional UDP traffic with datagrams of 1470 bytes, we need to start generating traffic at a low rate and increase it until the packet loss is above 1% or the delivered rate is below the generated one. The scenario for this test is shown in Figure 6.10. We run 100 second Iperf dualtests generating 10, 50, 100, 500, 1000, 5000, 10 000, 15 000, 20 000 and 25 000 Kbps bidirectional UDP streams for both IEEE 802.11a and SoftToken obtaining every second the values for the throughput, jitter and packet loss. Then we compute the mean and the standard deviation for the throughput for these values.
Plotting together the results for the throughput we find the upper bound on which SoftToken can deliver for each number of packets per queue and we compare it to the one achieved by IEEE 802.11a. The results for the uplink saturation throughput are shown in Figure 6.11. These results have shown that there are no significant differences between the UDP throughput in the uplink and the downlink. Hence we plot in Figure 6.11 the UDP throughput corresponding to the uplink.

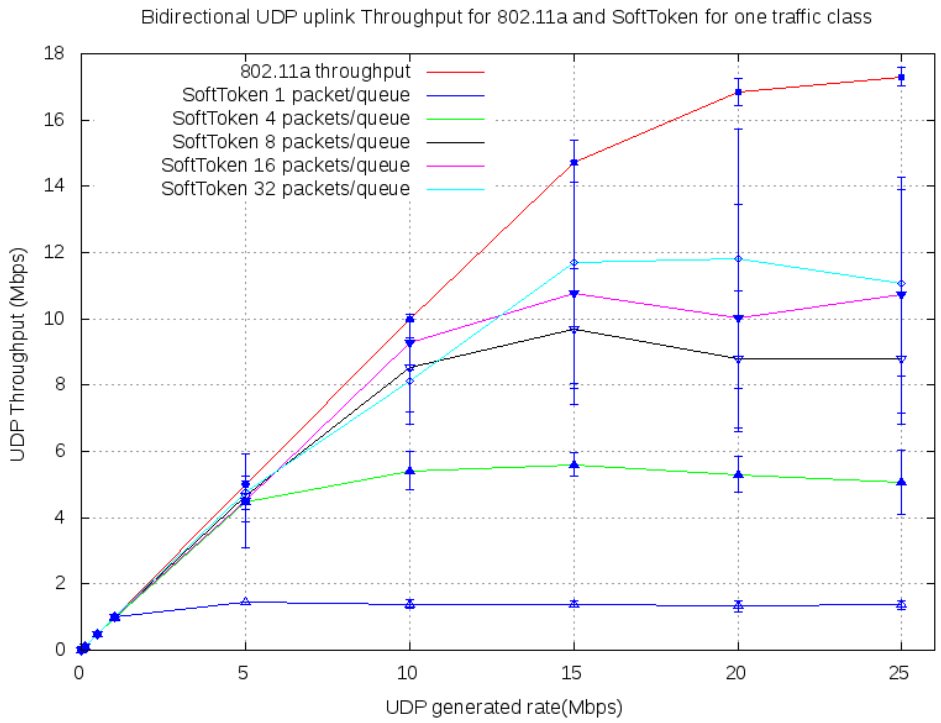


Figure 6.11: Bidirectional UDP uplink throughput for 802.11a and SoftToken at 54 Mbps

As it can be seen in Figure 6.11, IEEE 802.11a (the red line in the figure) has the highest UDP saturation throughput which is around 15 Mbps and the smallest error. For others, the delivered traffic is below the injected one.
For SoftToken, in Figure 6.11 we can see the evolution for the UDP throughput for 1, 4, 8, 16 and 32 packets per queue configurations. This way we measure the upper bounds for each of these configurations. As it was expected, in SoftToken, the UDP saturation throughput depends strongly in the number of packets per queue. We can see that the saturation throughput for 1 packet per queue is around 1 Mbps, however, for 4 packets per queue it increases until 5 Mbps

and for 8 packets per queue it is almost 10 Mbps. After that however, increasing the number of packets per queue does not increase significantly the saturation throughput. The absolute UDP saturation throughput for SoftToken is around 10 Mbps for 16 packets per queue. This value is again 5 Mbps under the one obtained for IEEE 802.11 making SoftToken unsuitable for this scenario.

In the two-node scenarios presented in this section the saturation throughput of SoftToken is far from improving the one of IEEE 802.11a. For the next section we choose only the resource allocations of SoftToken which provide a bidirectional UDP saturation throughput which is above 8 Mbps for 54 Mbps: 8, 16 and 32 packets for queue 0.

6.1.3.2 Bidirectional Saturation Throughput for 6, 36 and 54 Mbps Channel Data Rate

The objectives of the tests presented in this section are:

- Checking that in case of bidirectional UDP traffic with the same SoftToken allocation resource for the uplink and the downlink, the throughput ratio between the uplink and downlink is 1:1.
- Providing the results for the UDP saturation throughput, its error, the jitter and the packet loss for both the uplink and the downlink.
- Comparing the results obtained for SoftToken and IEEE 802.11a

We start providing the results for the uplink. The values for the UDP bidirectional saturation throughput in the uplink and its error are shown in Table 6.7 and they are plotted in Figure 6.12. We highlight the results for SoftToken corresponding to the resource allocation which gives better UDP saturation throughput.

Link data rate (Mbps)	**SoftToken thrp. for 8 pckt/q (Mbps)**	**SoftToken thrp. for 16 pckt/q (Mbps)**	**SoftToken thrp. for 32 pckt/q (Mbps)**	**802.11a thrp. (Mbps)**
6	1.940 ± 0.540	1.487 ± 0.475	2.368 ± 0.042	2.514 ± 0.042
36	11 ± 0	11.040 ± 0.070	11.115 ± 1.432	13.490 ± 0.197
54	10.178 ± 2.124	14.457 ± 1.833	15.270 ± 0.163	18.380 ± 0.282

Table 6.7: UDP bidirectional saturation throughput for SoftToken and IEEE 802.11a in the uplink

If we compare the results for SotToken and IEEE 802.11a in this scenario with the ones of the unidirectional test of the previous section, we see that now, in a bidirectional test, the difference between SoftToken and IEEE 802.11 has significantly decreased: from around $10Mbps$ of difference to around 3 Mbps we have now in the bidirectional test. This decrease in differ-

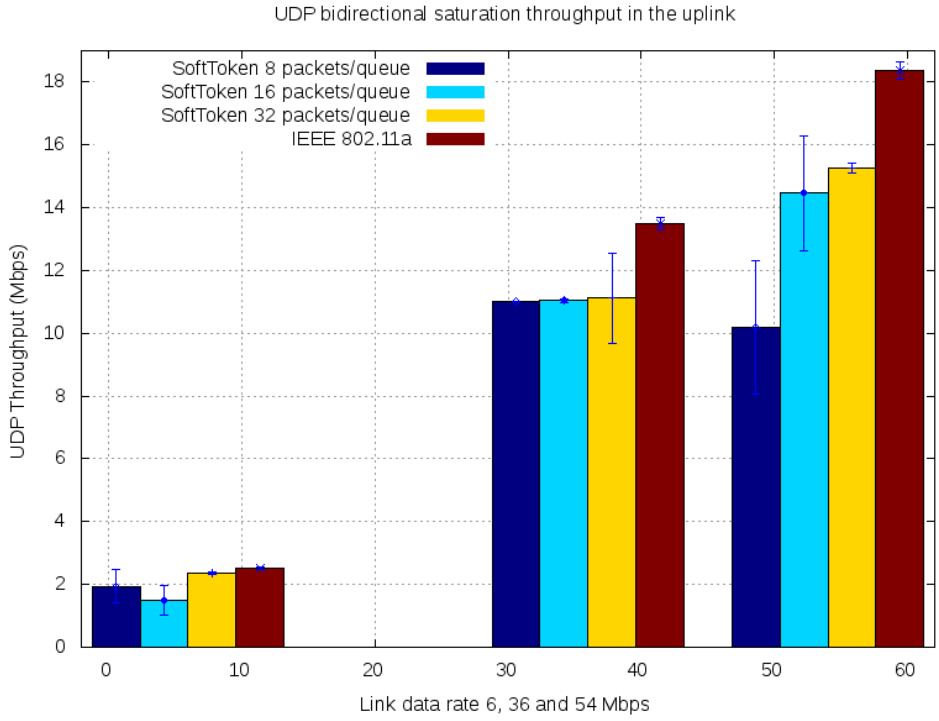


Figure 6.12: UDP bidirectional saturation throughput for SoftToken and IEEE 802.11a in the uplink

ence is due to the increase in contention for IEEE 802.11 for the bidirectional scenario. For the unidirectional one, there is no contention at all for 802.11 since only one station is transmitting. However, for SoftToken, even in the unidirectional test there is bidirectional management traffic as SoftToken management packets are sent between the master and the slave. Now, in the bidirectional test, IEEE 802.11 is subject to contention. However, SoftToken avoids it by applying its token mechanism. This explains the increase of the performance of SoftToken for this scenario with respect to IEEE 802.11.

Link data rate (Mbps)	**SoftToken thrp. for 8 pckt/q (ms)**	**SoftToken thrp. for 16 pckt/q (ms)**	**SoftToken thrp. for 32 pckt/q (ms)**	**802.11a thrp. (Mbps)**
6	8.031	8.269	7.140	9.946
36	1.972	2.079	1.949	1.810
54	1.705	1.405	1.511	1.065

Table 6.8: UDP average jitter in the uplink for SoftToken and IEEE 802.11a

The values for the average jitter in the uplink are shown in Table 6.8 and are plotted in Figure 6.13.

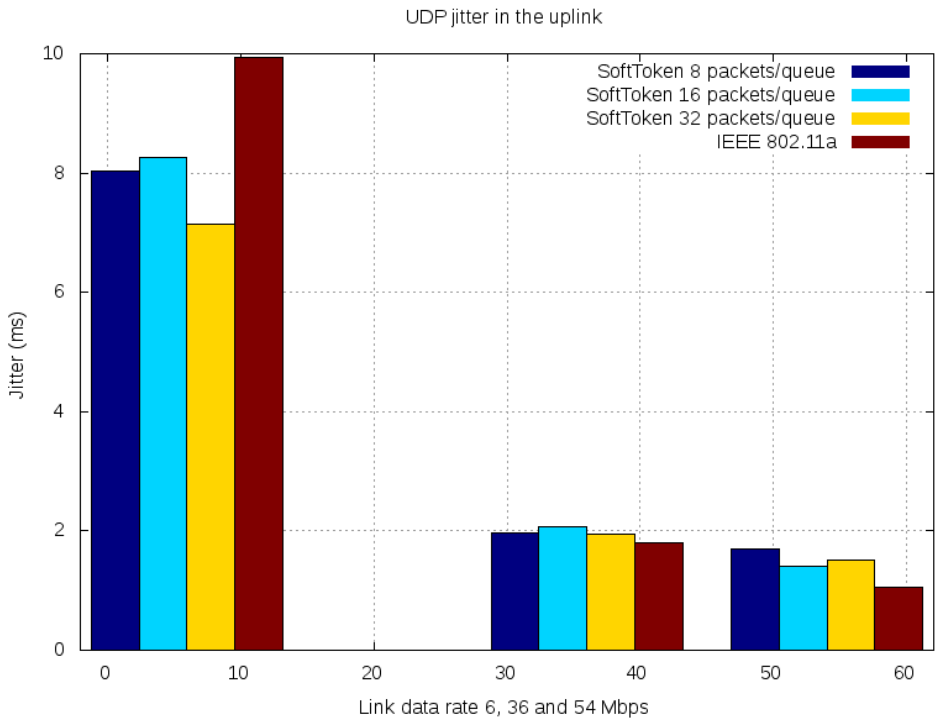


Figure 6.13: UDP average jitter in the uplink for SoftToken and IEEE 802.11a

As we can see, the values obtained for a physical rate of 6 Mbps due to buffered times, are unacceptable for VoIP applications, however, for 36 and 54 Mbps the jitter is between 1 and

$2ms$ which are more reasonable values.

The values for the average packet loss in the uplink are shown in Table 6.9 and are plotted in Figure 6.14.

Link data rate (Mbps)	**SoftToken pckt.loss loss 8 pckt/q (%)**	**SoftToken pckt. loss 16 pckt/q (%)**	**SoftToken pckt. loss 32 pckt/q (%)**	**802.11a pckt. loss (%)**
6	0.023	0.097	0	0.365
36	0.005	0.052	0.037	0.161
54	0.038	0.173	1.288	2.13

Table 6.9: UDP bidirectional average packet loss for SoftToken and IEEE 802.11a in the uplink

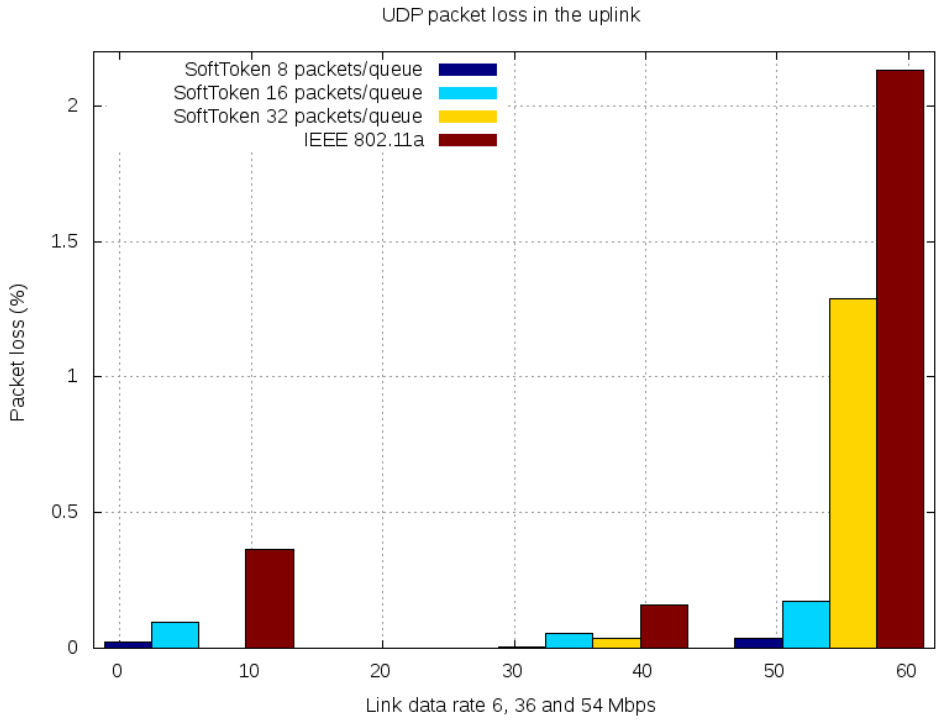


Figure 6.14: UDP packet loss in the uplink for SoftToken and IEEE 802.11a

As we can see, in general, the packet packet loss increases as the data rate increases. The values obtained for the packet loss are below 1% except for SoftToken with 32 packets per queue and IEEE 802.11 at 54 Mbps.

The values for the UDP bidirectional saturation throughput in the downlink and its error are shown in Table 6.10 and they are plotted in Figure 6.15. We highlight the results for SoftToken corresponding to the resource allocation which gives better UDP saturation throughput.

Link data rate (Mbps)	SoftToken thrp. for 8 pckt/q (Mbps)	SoftToken thrp. for 16 pckt/q (Mbps)	SoftToken thrp. for 32 pckt/q (Mbps)	802.11a thrp. (Mbps)
6	1.949 ± 0.529	1.527 ± 0.452	2.650 ± 0.030	2.633 ± 0.291
36	11 ± 0	10.970 ± 0.048	11.461 ± 1.425	13.160 ± 0.189
54	10.178 ± 2.097	14.126 ± 1.727	15.640 ± 0.271	17.560 ± 0.397

Table 6.10: UDP bidirectional saturation throughput for SoftToken and IEEE 802.11a in the downlink

As we can see, the results for the downlink are very similar to those for the uplink. We later provide a measure on how similar they are.

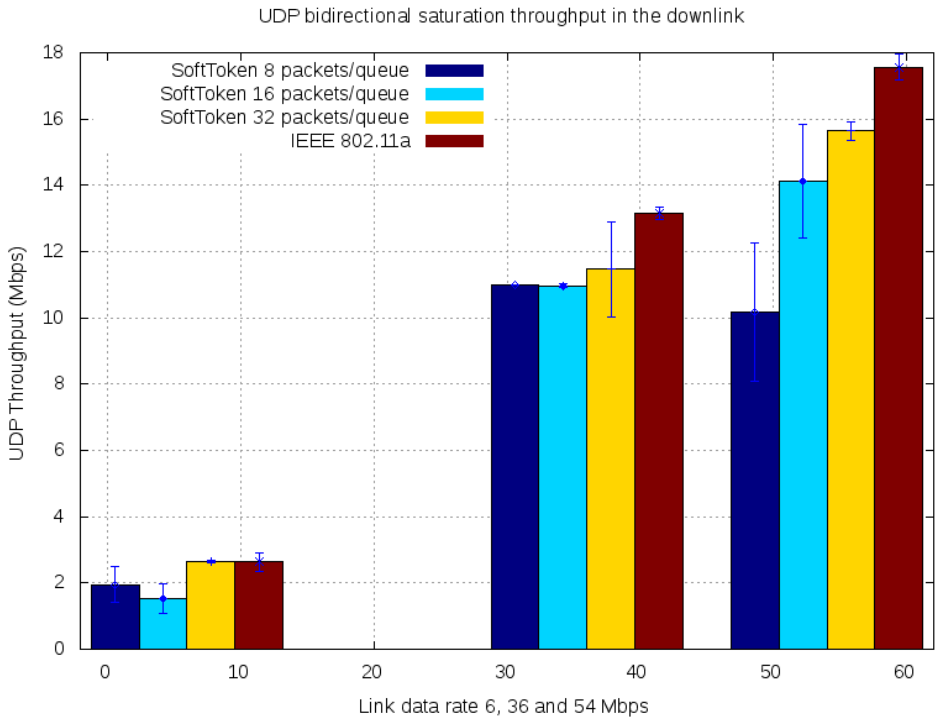


Figure 6.15: UDP bidirectional saturation throughput for SoftToken and IEEE 802.11a in the downlink

As we commented at the beginning of the section, we want to check that the UDP saturation throughput ratio between the uplink and the downlink for the same allocation resources in SoftToken is 1:1, so that there is no asymmetry between both channels. Dividing the results of Table 6.7 by the results of Table 6.10 and putting them in a form of ratios we find the values in Table 6.11. As we can see, all the ratios are around 1:1 which proves that SoftToken provides faireness in the transmission for the uplink and downlink when both are subject to the same resource allocation.

Link data rate (Mbps)	Ratio U:D for SoftToken for 8 pckt/q	Ratio U:D for SoftToken for 16 pckt/q	Ratio U:D for SoftToken for 32 pckt/q	Ratio U:D for 802.11a
6	1:1	0.97:1	0.89:1	0.95:1
36	1:1	1:0.99	0.97:1	1:0.97
54	1:1	1:0.98	0.97:1	1:0.95

Table 6.11: Ratio for the UDP throughput between the uplink and the downlink for SoftToken and IEEE 802.11a

6.1.4 Throughput in Two-node Bidirectional Mixed Traffic Tests

In the previous section we have checked that for the same resource allocation for the uplink and the downlink, SoftToken – as well as 802.11a – provided fairness in the transmission, that is, a ratio between the throughput in the uplink and the downlink of 1:1.
The objective of this test is to show the support for different TCs and priorities in SoftToken by allocating to different queues different bandwidths and to check that the ratio of the throughput corresponds to the ratio for the allocations. In particular, we are going to check a throughput ratio between high and low priority of 8:1 in a bidirectional scenario where the downlink has the low priority and the uplink has the high priority.
For the low priority downlink we assign to the queue 0, 4 packets – 5 880 bytes – and a type of service of 0. For the high priority uplink we assign to queue 3, 32 packets – 47 040 bytes – and a type of service of 3.
The configuration of the testbed is shown in Figure 6.16.

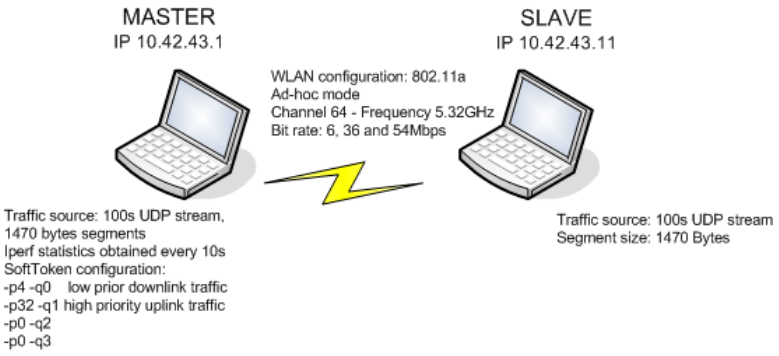


Figure 6.16: Configuration for the two-traffic classes bidirectional UDP test

Table 6.12 shows the results for the UDP saturation throughput and its error for the uplink and downlink for SoftToken and IEEE 802.11a.

In Figure 6.17 we see the difference between the throughput in the high priority uplink and the low priority downlink using SoftToken. On the other hand, IEEE 802.11a is not capable of traffic differentiation and so, both flows have similar throughputs. Moreover, we see that for

Link data rate (Mbps)	SoftToken thrp. low prior. downl. 4 p/q (Mbps)	SoftToken thrp. high prior. upl. 32 p/q (Mbps)	802.11a thrp. downl. (Mbps)	802.11a thrp. uplink (Mbps)
6	0.465 ± 0.374	2.229 ± 1.324	2.505 ± 0.031	2.516 ± 0.025
36	2.761 ± 0.013	16.3 ± 0.047	12.96 ± 0.189	13.01 ± 0.110
54	3.059 ± 0.285	18.4 ± 1.636	16.87 ± 0.305	17.87 ± 0.429

Table 6.12: UDP saturation throughput for SoftToken and IEEE 802.11a in the mixed traffic test

this case, SoftToken is outperforming IEEE 802.11 providing the high priority uplink higher throughput.

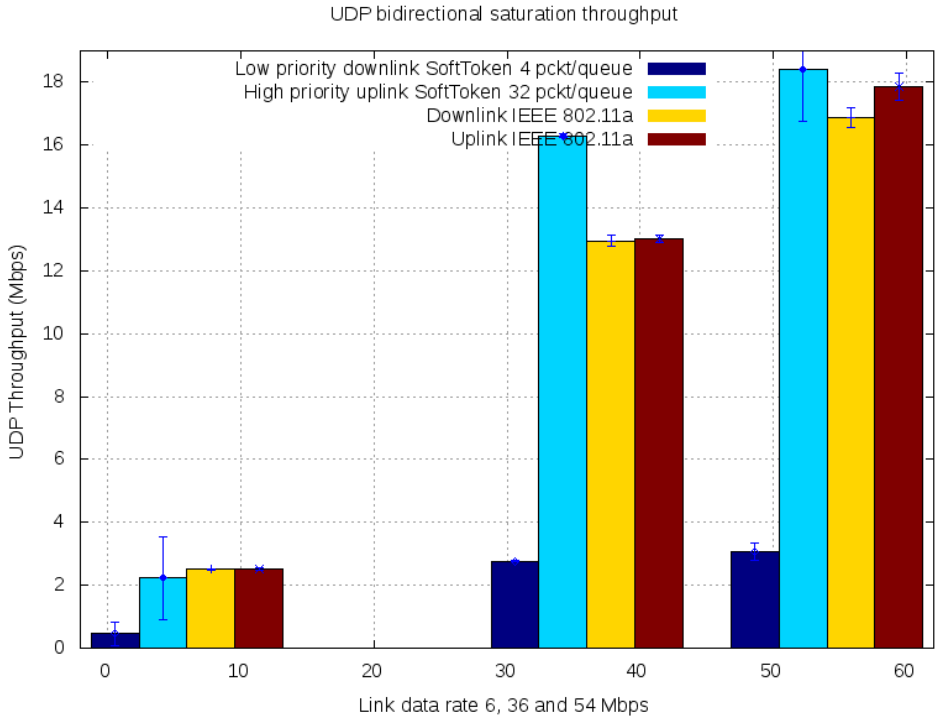


Figure 6.17: UDP saturation throughput in the uplink and the downlink for SoftToken and IEEE 802.11a in the mixed traffic test

As we did in the previous section, we are going to compute the ratio for the throughput between the high and the low priority to compare it to the resource allocation configured in SoftToken.

As we can see in Table 6.13, the ratio for the throughput between the high priority uplink and the low priority downlink is smaller than the one which is assigned by SoftToken. We obtain a ratio around 6:1 for a configured one of 8:1. Taking a look at Figure 6.17 however, we observe that this 8:1 ratio cannot be achieved because the channel gets saturated in the high priority uplink for 32 packets in queue 3 and not because SoftToken is not respecting the resource

Link data rate (Mbps)	Ratio Upl:Downl for SoftToken	Ratio Upl:Downl for 802.11a
6	4.8:1	1:0.99
36	5.9:1	1:0.99
54	6:1	1:0.94

Table 6.13: Ratio for the UDP throughput between the uplink and the downlink for SoftToken and IEEE 802.11a

allocation. This was proved in another testbed to conclude than in the absence of saturation, SoftToken respects the resource allocation it is assigned to a certain queue.

Table 6.14 and Figure 6.18 show the values for the average jitter for the low priority downlink and the high priority one.

Link data rate (Mbps)	SoftToken jitt. low prior. downl. 4 p/q (ms)	SoftToken jitt. high prior. upl. 32 p/q (ms)	802.11a jitt. downl. (ms)	802.11a jitt. uplink (ms)
6	74.489	16.92	7.938	7.056
36	8.408	1.142	1.317	1.430
54	6.444	1.038	1.063	0.9897

Table 6.14: UDP average jitter for SoftToken and IEEE 802.11a in the mixed traffic test

As we can see, the low priority downlink has higher values for the jitter than the uplink, which is expected due to the fact that the lower priority packets are longer queued.

Table 6.15 and Figure 6.19 show the values for the average packet loss for the low priority downlink and the high priority uplink.

Link data rate (Mbps)	SoftToken p.loss low prior. downl. 4 p/q (%)	SoftToken p.loss high prior. upl. 32 p/q (%)	802.11a p.loss downl. (%)	802.11a p.loss uplink (%)
6	0	0.046	0.009	0.255
36	0	0.758	0.240	0.474
54	0.056	1.207	2.31	3.37

Table 6.15: UDP average packet loss for SoftToken and IEEE 802.11a in the mixed traffic test

As we can see, in terms of packet loss, SoftToken performs better than IEEE 802.11 at 54 Mbps. This means that SoftToken reduces the contention when compared to IEEE 802.11.

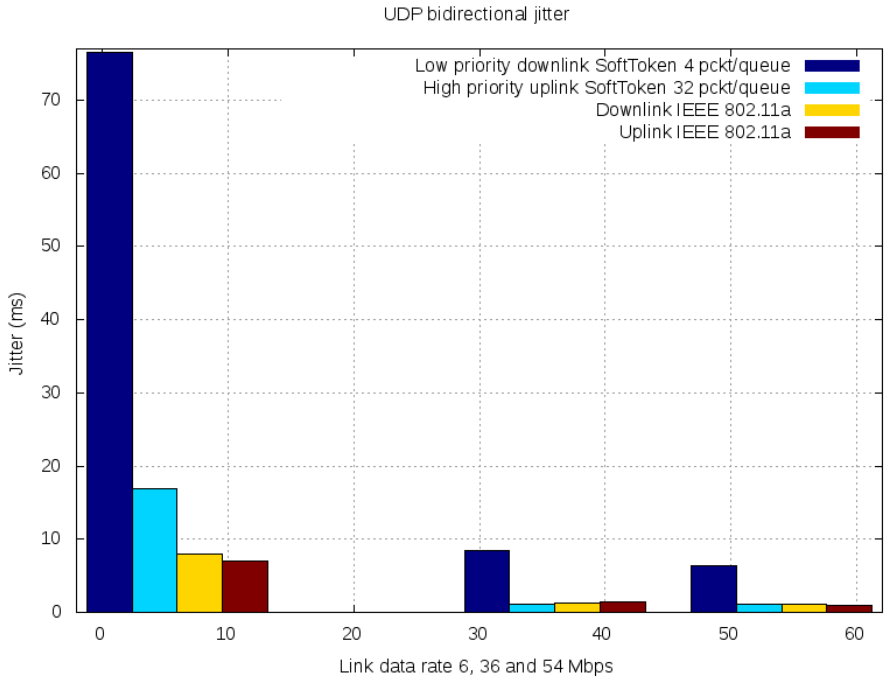


Figure 6.18: UDP average jitter in the uplink and the downlink for SoftToken and IEEE 802.11a in the mixed traffic test

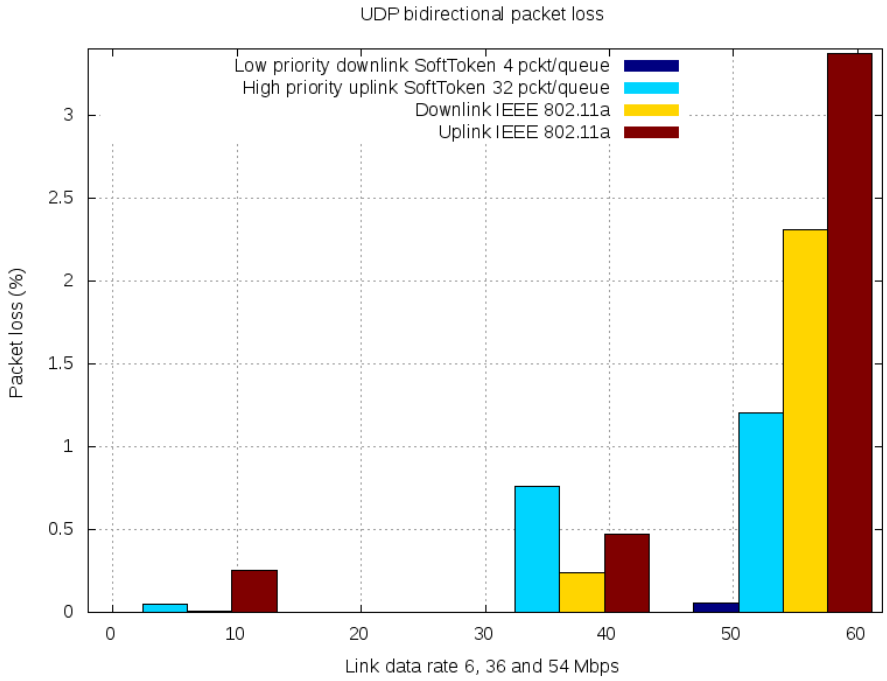


Figure 6.19: UDP average packet loss in the uplink and the downlink for SoftToken and IEEE 802.11a in the mixed traffic test

6.2 TCP Tests

6.2.1 Two-node Unidirectional Saturation Throughput

The objective of these tests is to find experimentally the TCP throughput for different parameter configurations for SoftToken in a two-node testbed with a channel rate of 54 Mbps. As the previous section, we also provide comparisons with IEEE 802.11a. To observe the evolution of the performance for these configurations we set up a testbed with just one unidirectional TCP stream sent from the slave to the master. Because of the random access to the medium used by 802.11a we need to run each test several times and calculate the mean and the standard deviation of the values obtained for the throughput. We will use these values to show the evolution and variation of the mean throughput. The configuration for the testbed is shown in Figure 6.20.

First, we obtain the TCP throughput that can be achieved using IEEE 802.11a for 54 Mbps.

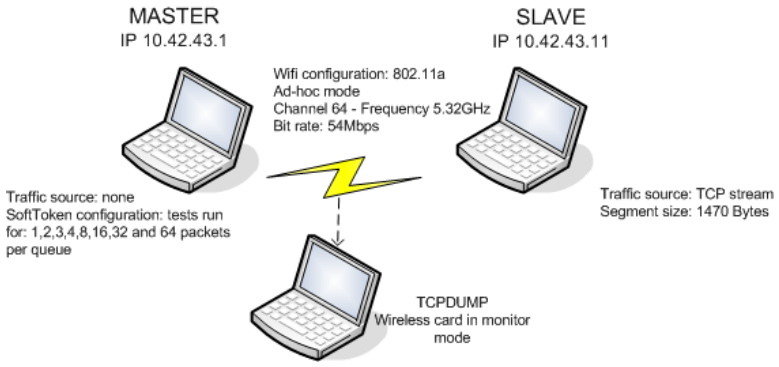


Figure 6.20: Testbed configuration for the TCP test

We run ten Iperf tests with the slave configured as the TCP client which establishes a TCP session with the master configured as the TCP server. Then, we compute the average and standard deviation of the values obtained for the throughput. Second, we obtain the maximum TCP throughput that can be achieved using SoftToken for different number of packets per queue. That is the number of packets that SoftToken allows to be sent one at a time. We run ten Iperf TCP throughput tests for each configuration of SoftToken corresponding to: 1, 2, 3, 4, 8, 16, 32 and 64 packets per queue and again we compute the average and standard deviation for these values.

The numerical results for the TCP throughput and its error obtained for SoftToken with different configurations are shown in Table 6.16 and plotted in Figure 6.21. The TCP throughput obtained for IEEE 802.11a is 28.27 Mbps and it corresponds to the constant green line of Figure 6.21. The error is equal to 0.551 Mbps. The reason that the mean TCP throughput appears as a constant is that it does not depend on the SoftToken configuration parameters.

Number of pckt/queue	TCP throughput (Mbps)	throughput error (Mbps)
1	1.242	0.227
2	2.731	0.305
3	3.879	0.446
4	5.157	0.814
8	10.667	1.140
16	15.529	3.638
32	18.870	2.970
64	15.996	4.354

Table 6.16: TCP throughput for an unidirectional stream and its error for SoftToken

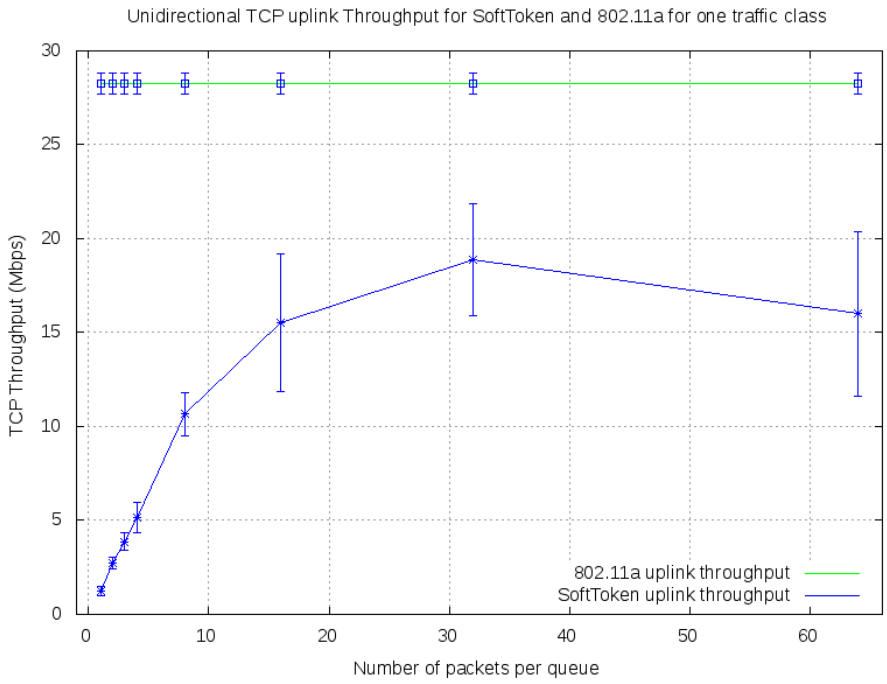


Figure 6.21: TCP throughput for 802.11a and SoftToken for different number of packets per queue for a channel data rate of 54 Mbps

The evolution of the throughput for SoftToken is depicted as a blue line and the errors are plotted with error bars. As it can be seen in Figure 6.21, the mean TCP throughput increases as the number of packets per queue increases and it reaches its maximum: 18.870 Mbps for 32 packets per queue. After 32 packets it starts decreasing. The difference in throughput with IEEE 802.11a is 9.4 Mbps which shows that SoftToken does not perform well in this scenario. With SoftToken the variation of the throughput is also higher, almost 8 times higher for the worst case, showing that it also increases the variation already introduced by IEEE 802.11a.

6.2.2 Throughput Comparison with SoftTDMAC in Two-node Tests

In this section, we present the results obtained for the TCP throughput tests for different channel rates and different configurations of SoftTDMAC.The configurations for SoftTDMAC and SoftToken are the same as the UDP tests show in section 6.1.2. To observe the evolution of the performance for different parameters we first set up a testbed with just one unidirectional TCP stream, that is, one traffic class, sent from the slave to the master. Again, we run each test several times and calculate the mean and the standard deviation. We use these values to show the evolution and variation of the TCP throughput. The configuration for the testbed is shown in Figure 6.22.

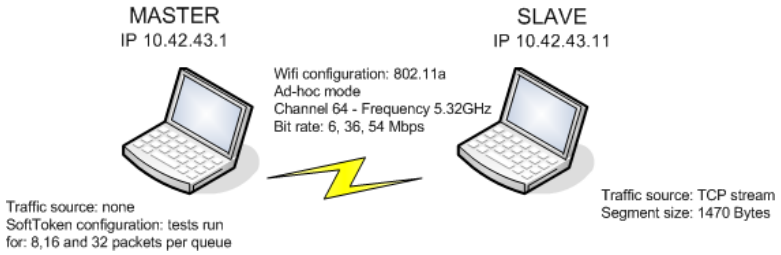


Figure 6.22: Configuration for the unidirectional TCP test

For these tests we used the channel 64. The direction of the data streams is uplink, from the Slave to the Master and the size of the data packets is 1470 bytes. For each configuration of SoftToken we ran 10 TCP tests, 60 seconds each. Then the average and standard deviation was computed.
Table 6.17 shows the results for the TCP throughput for different configurations of SoftTDMAC and Table 6.18 shows the ones for SoftToken and IEEE 802.11a . The best results for SoftTDMAC and SoftToken are highlighted in the tables. In order to compare the performance of SoftTDMAC, SoftToken and 802.11a we have plotted in Figure 6.23 the configuration of SoftTDMAC which achieves the highest throughput (20% Master and 70% Slave configuration), along with Table 6.18.

As in the UDP case, for the three channel data rates tested, 6, 36 and 54 Mbps, the performance of SoftToken for TCP is worse than the one for IEEE 802.11a. Comparing these values to the ones obtained for UDP – Table 6.2 and Table 6.3 – we can see that they are similar. The channel data rate that shows the best performance of SoftToken when compared to IEEE 802.11 is 6

Link data rate (Mbps)	SoftTDMAC thrp. for 20%M-70%S (Mbps)	SoftTDMAC thrp. for 45%M-45%S (Mbps)	SoftTDMAC thrp. for 70%M-20%S) (Mbps)
6	3.14	1.64	0.552
36	7.96	7.07	4.15
54	6.64	6.11	5.78

Table 6.17: TCP unidirectional throughput for SoftTDMAC

Link data rate (Mbps)	SoftToken thrp. for 8 pckt/q (Mbps)	SoftToken thrp. for 16 pckt/q (Mbps)	SoftToken thrp. for 32 pckt/q (Mbps)	802.11a thrp. (Mbps)
6	3.589 ± 0.292	3.806 ± 0.374	3.788 ± 0.398	4.663 ± 0.014
36	14.910 ± 2.247	15.720 ± 1.219	15.160 ± 1.813	22.060 ± 0.069
54	10.023 ± 0.761	16.850 ± 0.839	18.330 ± 2.126	29.070 ± 0.266

Table 6.18: TCP unidirectional throughput for SoftToken and IEEE 802.11a

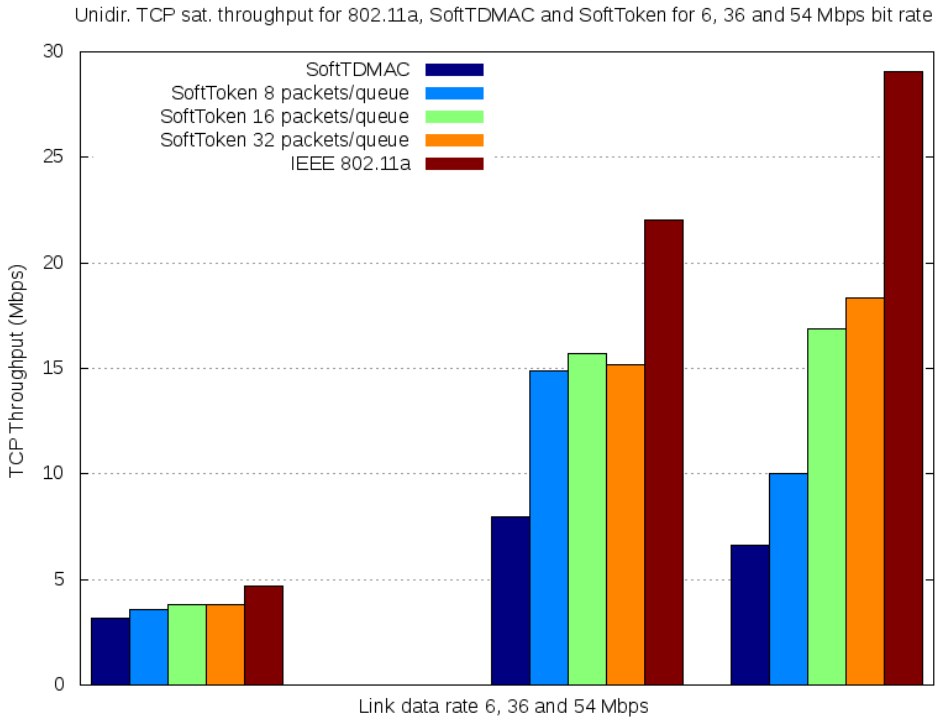


Figure 6.23: TCP unidirectional throughput for SoftTDMAC, SoftToken and IEEE 802.11a

Mbps, with a TCP throughput of 3.806 Mbps for SoftToken and 4.663 Mbps for IEEE 802.11a. This is because both data and control fields of the frames are sent at the same rate.

6.2.3 Analysis of the TCP Sequence Numbers in Two-node Unidirectional Tests

In order to explain the performance of SoftToken with TCP compared to IEEE 802.11, we plot the TCP sequence number versus time as well as the TCP packet throughput for a channel data rate of 6 Mbps, for the three configurations of SoftToken: 8, 16 and 32 packets for queue 0, in addition to IEEE 802.11a results corresponding to the tests of the previous section.
The figures show that for SoftToken, regularly TCP transmission is stopped. At first instance, we consider packet retransmissions. However, taking a closer look, we see it is not the case because no smaller sequence numbers, which means previous segments, are being retransmitted . Instead of that, TCP transmission is stopped for always the same duration: around 2 seconds which is the length of the retransmission timer for SoftToken. As it was explained in section 4.2.1.2, the SoftToken master assumes that the token has been lost if no response is received in 2 seconds. This explains the abnormal behaviour of the SoftToken TCP tests. When the SoftToken request is lost and is not received by the Slave, none of the nodes can transmit, as each one believes the token is owned by the other. So, for 2s, the TCP transmission is blocked. When the time out occurs in the Master, the SoftToken request is sent back to the Slave which can resume transmitting the TCP segments. As it is logical, in the case of IEEE 802.11 these intervals do not occur, the TCP sequence numbers increase continuously.
If we count the number of times the token is lost, we see that in average we have seven token retransmissions every 60 seconds, which roughly means that every 10 seconds the token is lost. This has a major effect on the performance, and it should be checked if it corresponds to a bug in the protocol.

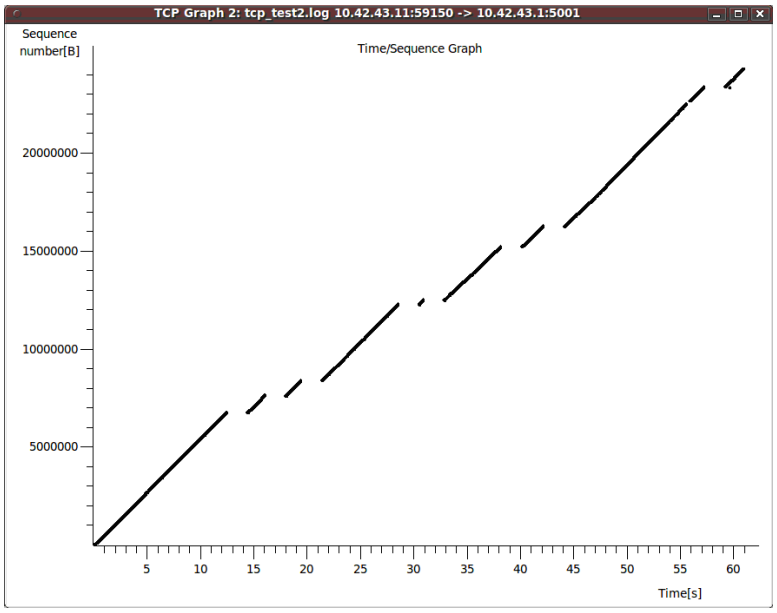


Figure 6.24: TCP sequence number vs time for a channel data rate of 6 Mbps and SoftToken with 8 packet/ queue

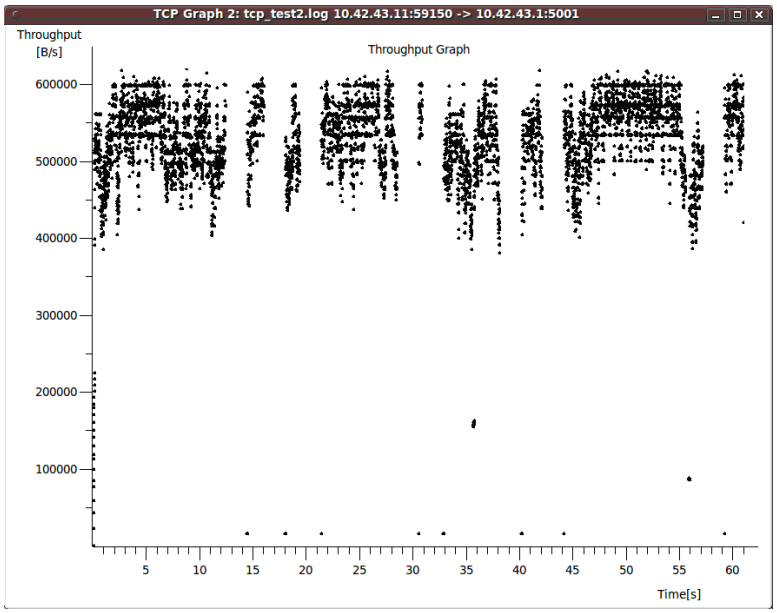


Figure 6.25: TCP packet throughput graph for a channel data rate of 6 Mbps and SoftToken with 8 packet/ queue

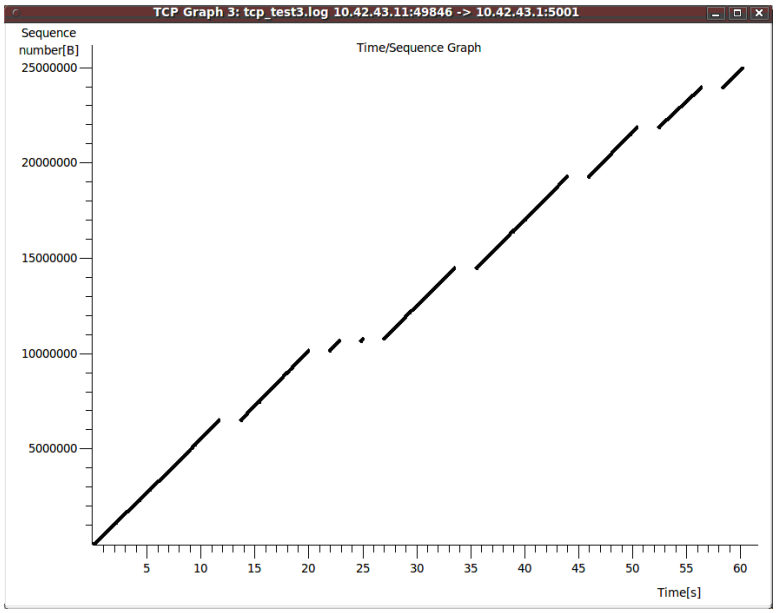


Figure 6.26: TCP sequence number vs time for a channel data rate of 6 Mbps and SoftToken with 16 packet/ queue

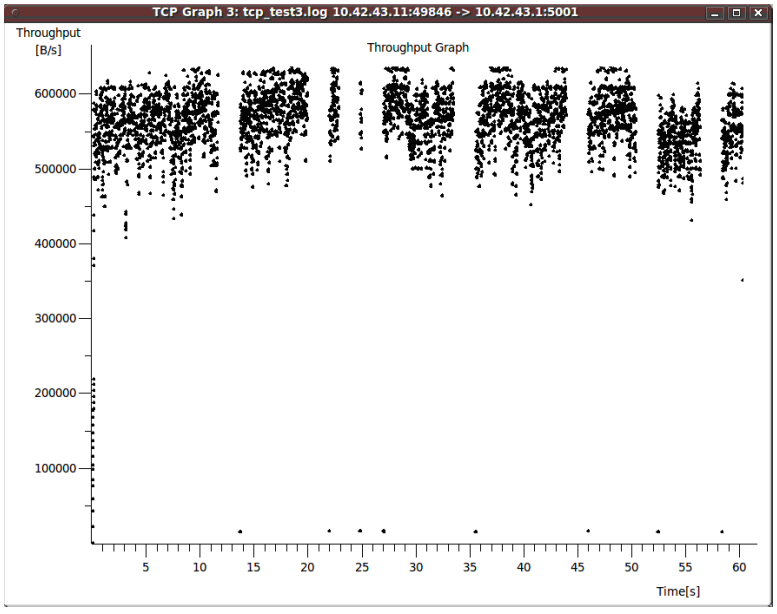


Figure 6.27: TCP packet throughput graph for a channel data rate of 6 Mbps and SoftToken with 16 packet/ queue

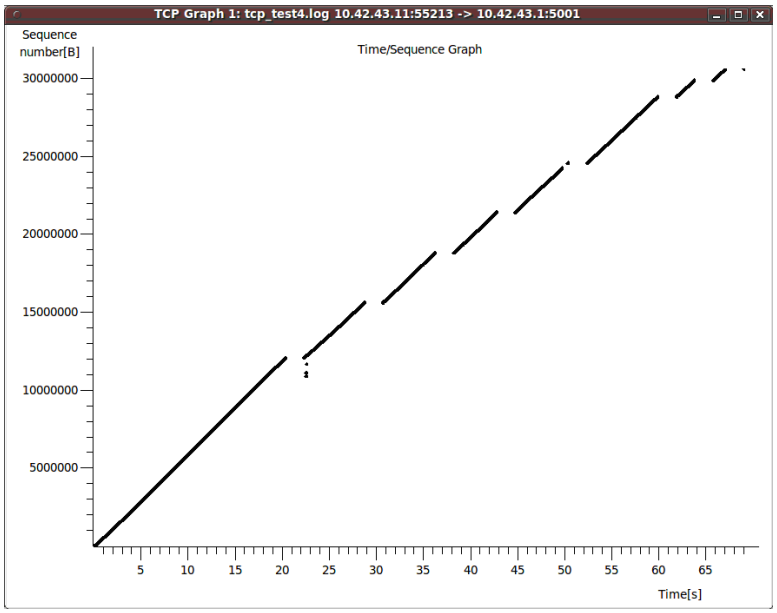


Figure 6.28: TCP sequence number vs time for a channel data rate of 6 Mbps and SoftToken with 32 packet/ queue

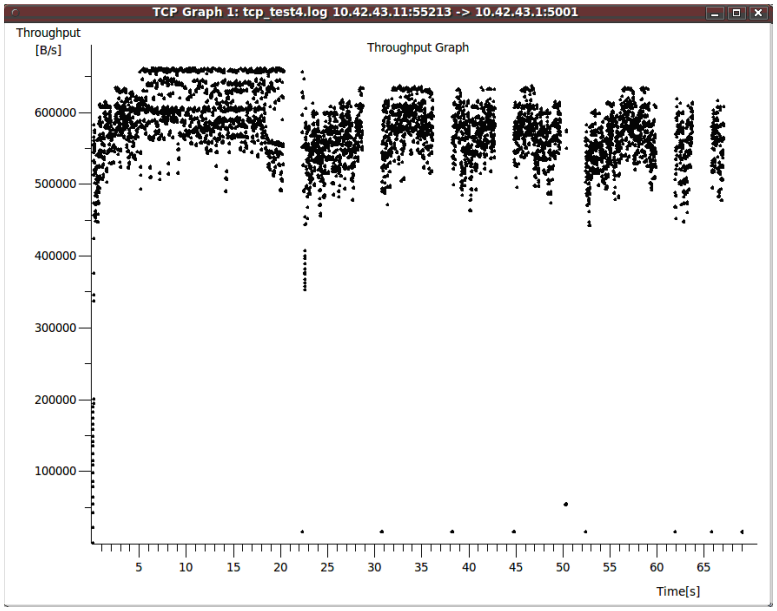


Figure 6.29: TCP packet throughput graph for a channel data rate of 6 Mbps and SoftToken with 32 packet/ queue

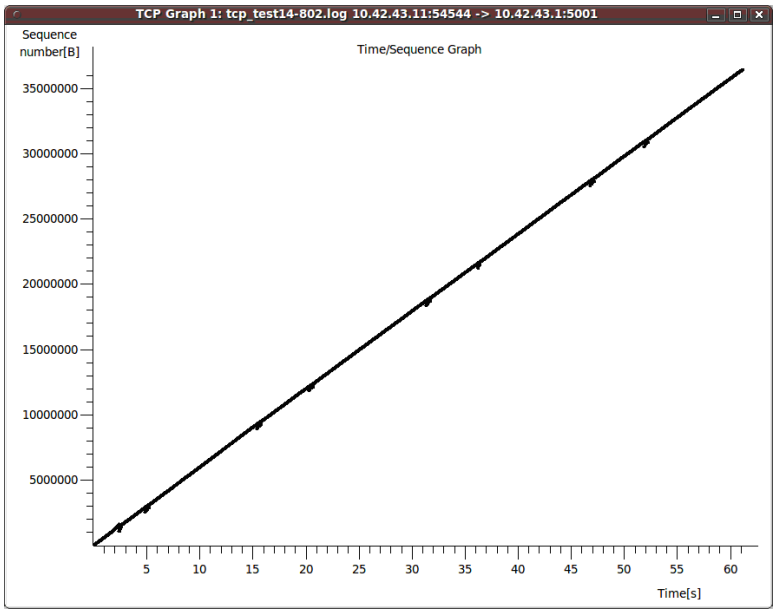


Figure 6.30: TCP sequence number vs time for a channel data rate of 6 Mbps for IEEE 802.11a

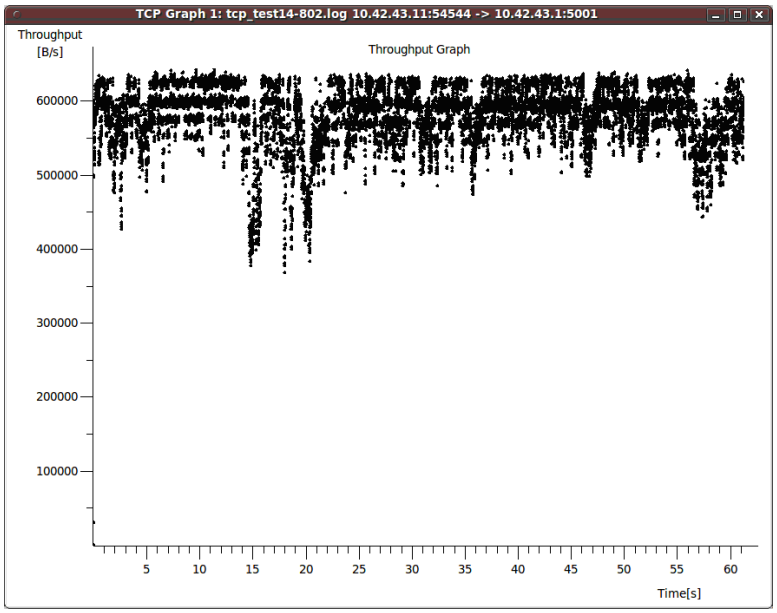


Figure 6.31: TCP packet throughput graph for IEEE 802.11a for a channel data rate of 6 Mbps

A way to prove that the interruptions in the transmission are caused by a loss of a token management packet consist of increasing the value of the retransmission timer to an extreme value. In this case, we choose to increase the retransmission timer of SoftToken from 2 seconds to 8 seconds and to run again the tests for SoftToken. The results are shown from Figure 6.32 to Figure 6.35.
As it can be appreciated, the intervals for which the transmission is stopped have increased with respect to the previous tests from 2 seconds to 8 seconds. This, effectively proves that this interruption is caused by a SoftToken management packet loss, which in turn is the reason for the decreasing of the performance of SoftToken. Therefore it is necessary to research on what originates the token losses as well as to choose a more accurate value for the retransmission timer to have a better trade-off between performance and reliability.

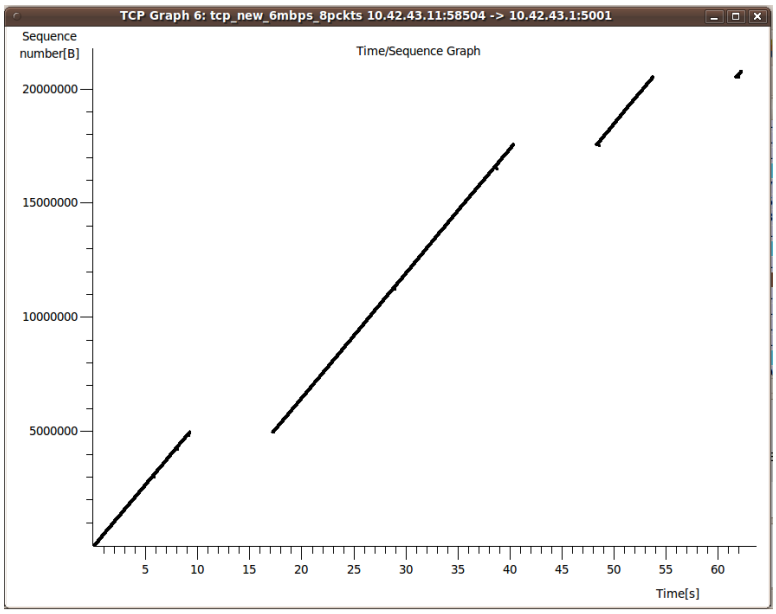


Figure 6.32: TCP sequence number vs time for 6Mbps and SoftToken with 8 packet/queue for a retransmission timer of 8 seconds

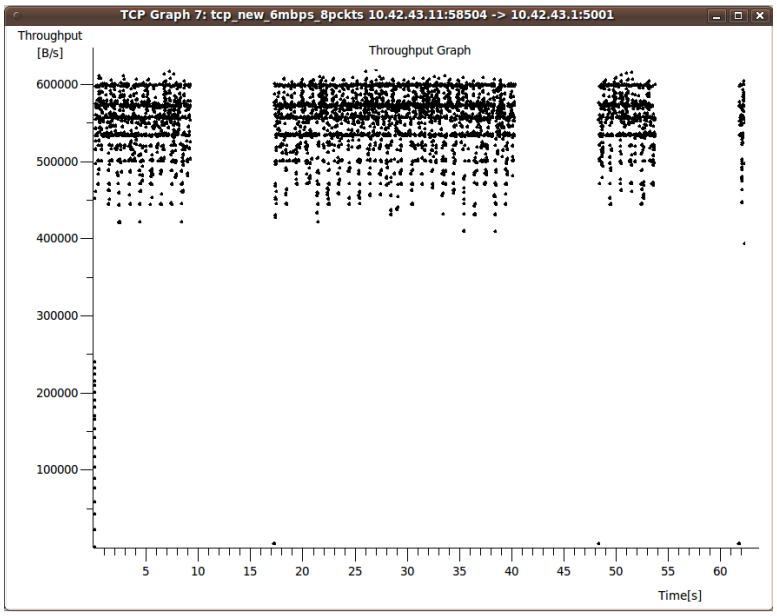


Figure 6.33: TCP packet throughput graph for 6Mbps and SoftToken with 8 packet/queue for a retransmission timer of 8 seconds

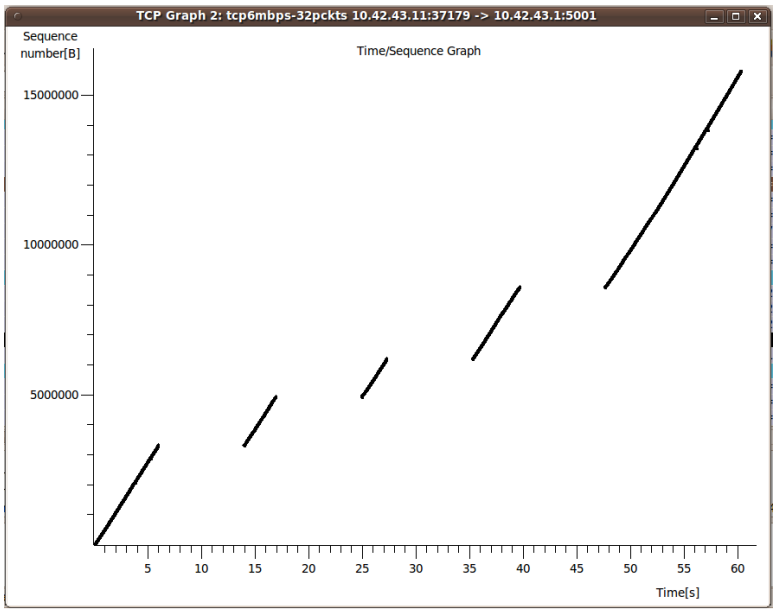


Figure 6.34: TCP sequence number vs time for 6Mbps and SoftToken with 32 packet/queue for a retransmission timer of 8 seconds

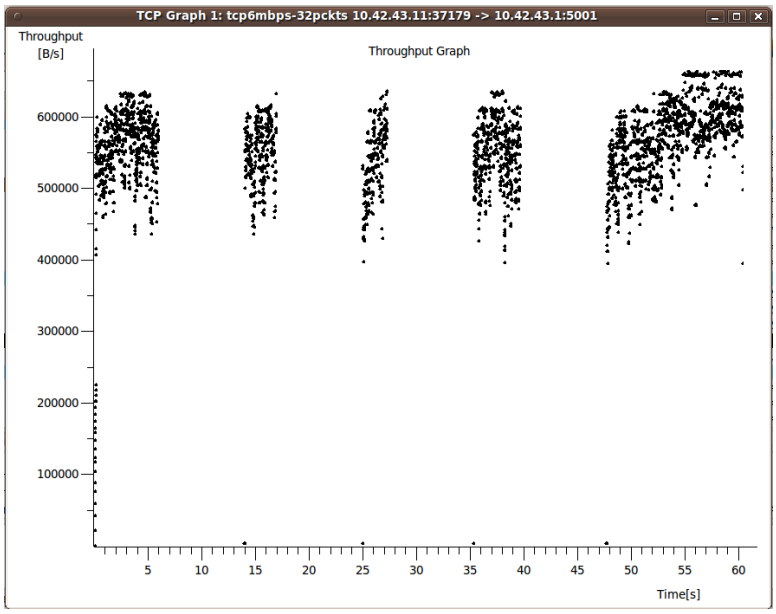


Figure 6.35: TCP packet throughput graph for 6Mbps and SoftToken with 32 packet/queue for a retransmission timer of 8 seconds

6.2.4 Two-node Bidirectional Throughput

6.2.4.1 TCP Throughput for Different Resource Allocations

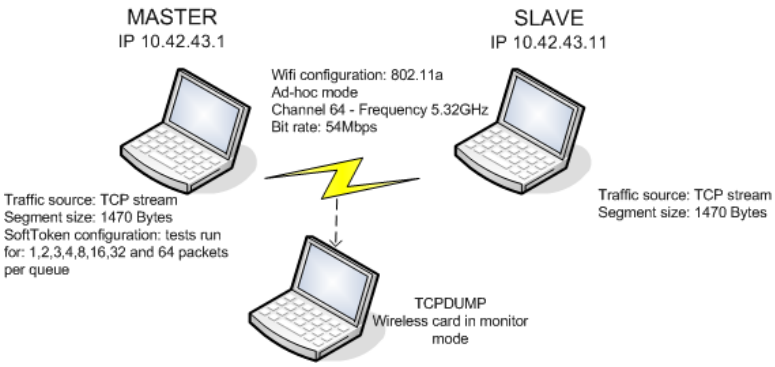


Figure 6.36: Testbed configuration for the TCP test

As we did for the UDP tests, once the TCP throughput has been obtained for a unidirectional TCP stream, we want to observe which is the mean TCP throughput for both the uplink and downlink when we have simultaneous bidirectional communication. Figure 6.36 shows the configuration used in the testbed. Using Iperf in dualtest mode we measure bidirectional bandwidth simultaneously. The uplink refers to the communication taking part from the slave to the master.
As for the previous testbeds, we first run ten bidirectional tests for IEEE 802.11a and we compute the mean and the standard deviation for the uplink and downlink TCP throughput. Then,

for each SoftToken configuration, we also run ten tests and we perform the same calculations. The numerical results for the mean TCP throughput and its error for SoftToken with different configurations are shown in 6.19 and plotted in Figure 6.37.

Number of pckt/queue	TCP sat. thrp. uplink (Mbps)	sat. thrp. error uplink (Mbps)	TCP sat. thrp. downlink (Mbps)	sat. thrp. error downlink (Mbps)
1	0.753	0.210	0.695	0.222
2	1.597	0.544	1.588	0.222
3	2.256	0.857	2.398	0.757
4	3.361	0.742	2.924	0.538
8	7.320	0.933	6.086	1.352
16	12.376	2.958	9.756	1.244
32	11.758	3.463	9.073	2.485
64	11.560	4.234	9.464	2.041

Table 6.19: Uplink and downlink TCP throughput and its error for SoftToken

The uplink mean TCP throughput is represented by a black line and the errors are plotted with error bars. The downlink mean TCP throughput is represented by a red line.

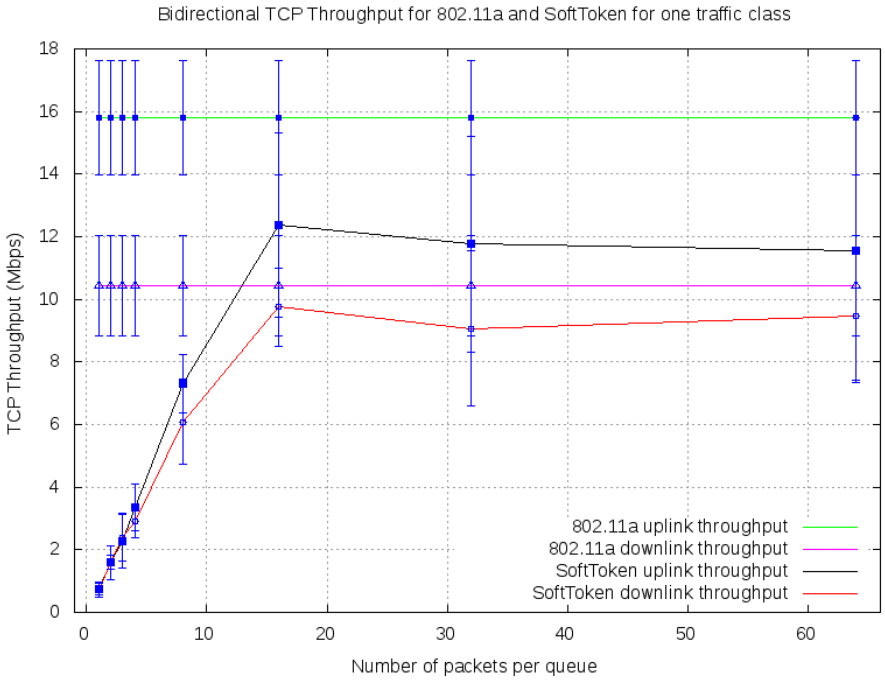


Figure 6.37: TCP throughput in the uplink and downlink for 802.11a and SoftToken at 54 Mbps

The first remarkable result is the high asymmetry in the TCP throughput between the uplink and the downlink for both SoftToken and IEEE 802.11a which is more severe in the case of IEEE 802.11a. This is due to the dualtest performed with Iperf in which the first TCP connection which is established, that is the one for the uplink, gets more bandwith than the reverse one.

In this case, the number of packets per queue that achieves the highest throughput in both directions is 16 reaching 12.376 Mbps for the uplink and 9.756 Mbps for the downlink. Beyond that, TCP throughput degrades because of waiting for an ACK to continue sending packets. Now, the difference between SoftToken and IEEE 802.11a has again decreased with respect to the TCP unidirectional test as it occured for the UDP tests. This difference is 3.424 Mbps for the uplink and 0.674 Mbps for the downlink taking closer SoftToken and IEEE 802.11a.

6.2.4.2 TCP Throughput for 6, 36 and 54 Mbps Channel Data Rate

As we did in the case of the bidirectional UDP tests, we now want to provide the results for the TCP throughput, its error, the jitter and the packet loss for both the uplink and the downlink and to compare them to the ones of IEEE 802.11a.

We start providing the results for the uplink. The values for the TCP throughput in the uplink for the bidirectional test and its error are shown in Table 6.20 and they are plotted in Figure 6.38. We highlight the results for SoftToken corresponding to the resource allocation which gives better TCP throughput in the uplink.

Link data rate (Mbps)	**SoftToken thrp. for 8 pckt/q (Mbps)**	**SoftToken thrp. for 16 pckt/q (Mbps)**	**SoftToken thrp. for 32 pckt/q (Mbps)**	**802.11a thrp. (Mbps)**
6	1.575 ± 0.735	1.699 ± 0.807	2.339 ± 0.701	2.700 ± 0.234
36	7.967 ± 0.288	11.010 ± 0.369	11.810 ± 0.731	13.610 ± 0.628
54	7.320 ± 0.932	12.376 ± 2.958	11.758 ± 3.462	15.810 ± 1.820

Table 6.20: TCP bidirectional throughput and error for SoftToken and IEEE 802.11a in the uplink

We provide now the results for the downlink. The values for the TCP throughput in the downlink for the bidirectional test and its error are shown in Table 6.21 and they are plotted in Figure 6.39. We highlight the results for SoftToken corresponding to the resource allocation which gives better TCP throughput in the downlink.

Link data rate (Mbps)	**SoftToken thrp. for 8 pckt/q (Mbps)**	**SoftToken thrp. for 16 pckt/q (Mbps)**	**SoftToken thrp. for 32 pckt/q (Mbps)**	**802.11a thrp. (Mbps)**
6	1.454 ± 0.428	1.475 ± 0.549	1.946 ± 0.450	2.110 ± 0.249
36	6.347 ± 0.296	8.618 ± 0.393	8.874 ± 0.880	9.453 ± 0.661
54	6.086 ± 1.352	9.756 ± 1.244	9.073 ± 2.485	10.430 ± 1.601

Table 6.21: TCP bidirectional throughput and error for SoftToken and IEEE 802.11a in the downlink

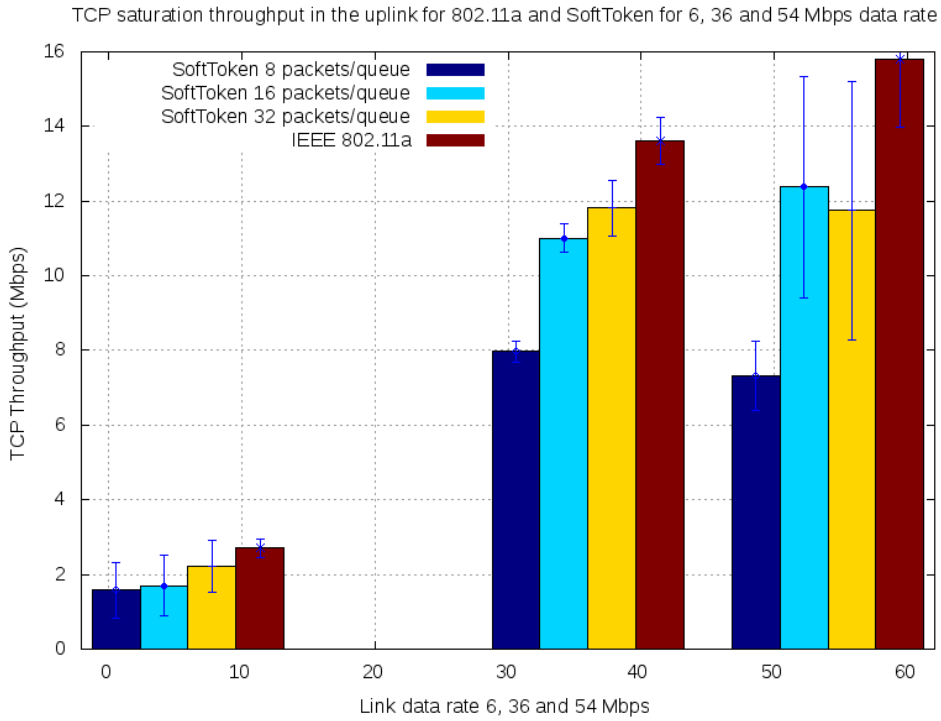


Figure 6.38: TCP bidirectional throughput for SoftToken and IEEE 802.11a in the uplink

Again, if we compare the results for SotToken and IEEE 802.11a in this scenario with the ones of the TCP unidirectional test of the previous section, we see that the difference between SoftToken and IEEE 802.11 has significantly decreased due to the increase in the contention in IEEE 802.11a generated by the bidirectional flows.

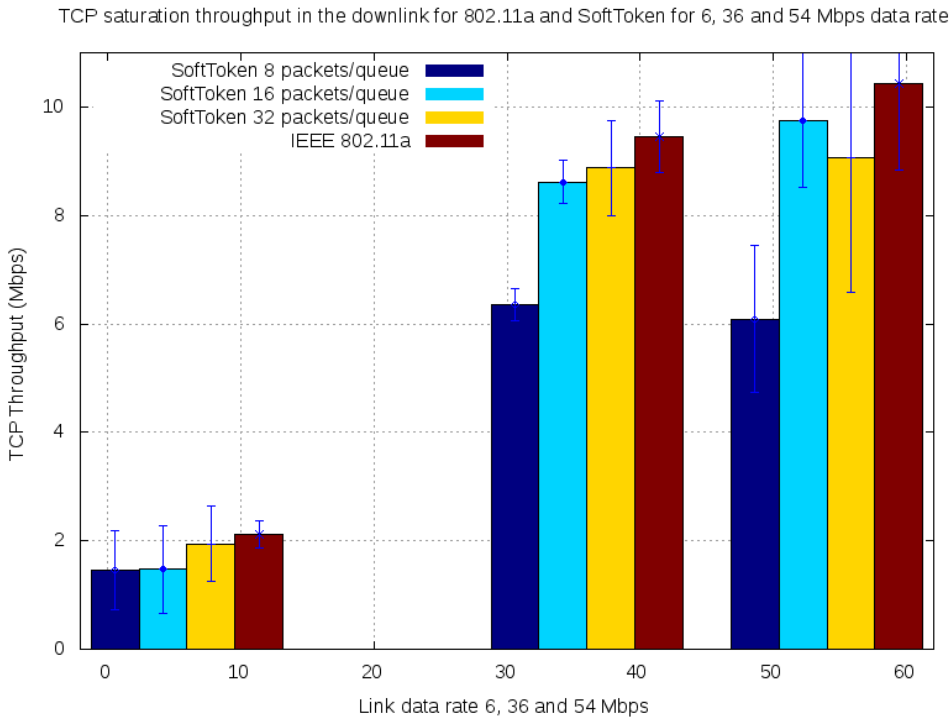


Figure 6.39: TCP bidirectional throughput for SoftToken and IEEE 802.11a in the downlink

Chapter 7

Conclusion and Future Work

In this chapter, we present the conclusion and a summary about the performance tests. Next, we offer some suggestions for future work.

7.1 Conclusion

Due to the novelty of SoftToken there is no available documentation on it. The explanation of the SoftToken protocol that it is provided tries to give a first approach to its functionality as well as to its architecture. In addition, no operational bounds neither theoretical nor practical are available. This is the reason for developing a simple theoretical analysis as well as for carrying out experimental performance tests.
The first steps towards the integration of SoftToken into the CARMEN framework have been presented as a part of the CARMEN project. First, the CARMEN self-configuration function is able to set up and configure a Link Group in order to be managed by SoftToken. Next, resource allocations in the Link Group can be requested, modified and released, respecting the QoS requirements which are granted by a dynamic scheduler implemented in the SoftToken Link Group Controller.

7.2 Summary of Contributions and Performance Evaluation

Our performance evaluation was beneficial to understand the limitations of the SoftToken protocol and determine the source of problems.
In summary, the highest UDP saturation throughput for SoftToken is achieved for only one resource allocation of 32 packets in queue 0 obtaining:

- 18.01 Mbps in the two-node unidirectional test against 27.16 Mbps of IEEE 802.11a.
- 15.270 Mbps in the uplink and 15.640 Mbps in the downlink in the two-node bidirectional test for a 1:1 uplink:downlink resource allocation against 18.380 Mbps and 17.560 Mbps of IEEE 802.11a.

The maximum TCP throughput for SoftToken is:

- 18.330 Mbps in the two-node unidirectional test for an allocation of 32 packets to queue 0 against 29.070 Mbps of IEEE 802.11a.
- 12.376 Mbps in the uplink and 9.756 Mbps in the downlink in the two-node bidirectional test for a 1:1 uplink:downlink resource allocation against 15.810 Mbps and 10.430 Mbps of IEEE 802.11a.

From these results we can conclude that, in the absence of traffic differentiation, SoftToken is not useful in a two-node scenario when compared with IEEE 802.11a. However, we have to remark that for the case of bidirectional scenarios, in which contention has increased, SoftToken throughtput gets closer to the one of IEEE 802.11a. This points out to the possibility that in a higher-contention scenario, the throughput of SoftToken could outperform the one of IEEE 802.11 even in the absence of traffic differentiation. This is left as future work.
The biggest contribution of SoftToken is the capacity of efficiently supporting traffic differentiation, respecting the ratio between the resource allocations which is critical for QoS-sensitive applications.
If we now compare the difference of performance between the theoretical model proposed in section 5.3 and the experimental results of section 6.1, we quickly realize there is something wrong, either in the model, or in the implementation of SoftToken or even in both.
In section 6.2.3 we proved that the token was lost at a rate of one loss every 10 seconds, having to wait 2 seconds after each loss to a retransmission time out to occur and resume the transmission. This is causing a reduction of 20% in the performance of SoftToken in respect to the model which is still far from explaining the differences. If such an unexpected behaviour is occuring it is likely that others are also happening, such as the bug that was found and solved as explained in section 6.1.1.1. This makes us conclude that, in parallel with the model revision, a careful debugging of SoftToken has to be done in order to improve its performance.

7.3 Future Work

Here we point out some suggestions for future work.

- Review the SoftToken implementation in order to solve the problem of the SoftToken management packet losses so as to improve its performance and validate the theoretical model. Every time a SoftToken management packet *softtoken_mngt_request* or *softtoken_mngt_response* is lost, the communication is interrupted as each node believes the token is owned by some of the other nodes and we have to wait for a retransmission time out in the master to resume the communication.

- Find the value for the retransmission timer which gives a better trade-off between performance and reliability. If the value of the retransmission timer is too high, as it is currently the case (2 seconds), the transmission is interrupted in this interval decreasing the performance. However, if the value is too small, the master can interpret that a SoftToken management packet has been lost when in fact is not so, thus retransmitting a *softtoken_mngt_request* that could collide either with a data packet or a *softtoken_mngt_response*. This could affect the reliability of SoftToken and violate the token principle.

- Design and run new tests for SoftToken for more nodes and more traffic classes. In scenarios with more than two nodes, the contention in IEEE 802.11 will increase. Therefore, it would be interesting to run SoftToken in three and four node-tests and to compare the results with the ones obtained for IEEE 802.11. It is likely that in these scenarios SoftToken will outperform IEEE 802.11.

- Develop new schedulers to support per-link resource allocations instead of per-Link Group resource allocations which is the case at present. In this situation, the maximum number of different resource allocations that can exist simultaneously in the scheduler is reduced to the number of queues which is currently 4. These allocations are shared by all the nodes which form the Link Group. However, the traffic class and bandwidth requirements for each link of the Link Group might be different, this is the reason to support per-link resource allocation instead of sharing it among the Link Group.

Bibliography

[1] I. Akyildiz and X. Wang, *Wireless Mesh Networks*, 1st ed. Wiley, 2009.

[2] D1.1, *First Architecture Deliverable*. CARMEN Deliverable, Aug. 2008.

[3] D2.5, *Final Report on Implementation and Simulation of WP2 Architecture*. CARMEN Deliverable, July 2010.

[4] N. Bayer and K. Loziak and A. García-Saavedra and C. Sengul and P. Serrano, "CARMEN: Resource Management and Abstraction in Wireless Heterogeneous Mesh Networks," in *SIGCOMM 2010*, New Delhi, India, Sept. 2010.

[5] D2.4, *Specification and Analysis of Media Access Control Mechanisms*. CARMEN Deliverable, Dec. 2009.

[6] B. Gloss, "WP2 Results 2009 - CARMEN 2nd Technical Audit," Alcatel-Lucent, Tech. Rep., 2010.

[7] *IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std. 802.11, 2007.

[8] P. Frank, "Advanced Transmission Schemes for the Fourth Generation of Cellular Networks," Ph.D. dissertation, Universität Stuttgart, 2010.

[9] M. Kuran and T.Tugcu, "A survey on emerging broadband wireless access technologies," *Computer Networks*, vol. 51, pp. 3013–3046, Aug. 2007.

[10] N. Bayer and A. Roos and R. P. Karrer and B. Xu and C. Esteve, "Towards Carrier Grade Wireless Mesh Networks for Broadband Access," in *First IEEE International Workshop On Operator-Assisted (Wireless Mesh) Community Networks*, Sept. 2007.

[11] (2010, Sept.) NGN working definition. International Telecommunication Union. [Online]. Available: http://itu.int/ITU-T/studygroups/com13/ngn2004/working_definition.html

[12] E. Hosain and K. Leung, *Wireless Mesh Networks Architectures and Protocols*, 1st ed. Springer, 2008.

[13] R. Ramanathan and J. Redi, "A brief overview of ad hoc networks: challenges and directions," *Communications Magazine, IEEE*, vol. 40, pp. 20–22, May 2002.

[14] G. Aggelou, *Mobile Ad Hoc Networks, From Wireless LANs to 4G Networks*, 1st ed. McGraw Hill Professional Engineering, 2004.

[15] I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks (Elsevier)*, vol. 47, pp. 445–487, Jan. 2005.

[16] (2010, Sept.) Microsoft Research. [Online]. Available: http://research.microsoft.com/en-us/projects/mesh/

[17] (2010, Sept.) Meraki. [Online]. Available: http://meraki.com/products_services/overview/

[18] (2010, Sept.) Strix Systems. [Online]. Available: http://www.strixsystems.com/solutions.aspx

[19] M. Hassan-Ali, "Municipal Wireless Mesh Networks as a Competitive Broadband delivery Platform," Master's thesis, MIT, Feb. 2007, thesis for the Degree of Master of Science in Engineering and Management.

[20] (2010, Sept.) MIT Roofnet. [Online]. Available: http://pdos.csail.mit.edu/roofnet/

[21] (2010, Sept.) Berlin Freifunk. [Online]. Available: http://berlin.freifunk.net/

[22] (2010, Sept.) List of wireless community networks by region. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/List_of_wireless_community_networks_by_region

[23] (2010, Sept.) Berlin Open Wireless Lab (BOWL). Intelligent Networks (INET) group at Deutsche Telekom Laboratories. [Online]. Available: http://bowl.net.t-labs.tu-berlin.de/

[24] (2010, Sept.) CARMEN: CARrier grade MEsh Networks. EU Seventh Framework Programme. [Online]. Available: http://www.ict-carmen.eu/

[25] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications-Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements*, IEEE Amendment 802.11e, 2005.

[26] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications-Draft amendment: ESS mesh networking*, IEEE Draft 802.11s, 2006.

[27] *Local and metropolitan area networks-Part 5 Token Ring Access Method and Physical Layer Specification*, IEEE Std. 802.5, 1998.

[28] Cisco, *Internetworking Technologies Handbook - Chapter 9*, 4th ed. Cisco Press, 2003.

[29] A. S. Tanenbaum, *Computer Networks*, 4th ed. Prentice Hall, 2003.

[30] M. Ergen, D. Lee, A. Puri, P. Varaiya, R. Attias, R. Sengupta, and S. Tripakis, "Wireless Token Ring Protocol," *IEEE Transactions on Vehicular Technology*, vol. 56, pp. 1863–1881, Nov. 2004.

[31] D2.2, *Final Assessment of MAC Layer Abstraction*. CARMEN Deliverable, May 2009.

[32] *Draft Standard for Local and Metropolitan Area Networks: Media Independent Handover Services*, IEEE Working Draft Proposed Standard 802.21, 2008.

[33] P. Djukic and P. Mohapatra, "Soft-TDMAC: Software TDMA-based MAC over commodity 802.11 hardware," *INFOCOM 2009. The 28th Conference on Computer Communications*, pp. 1836–1844, Apr. 2009.

[34] N. Bayer and C. Sengul, "CARMEN DTAG Coordinated MAC status," Deutsche Telekom Laboratories, Tech. Rep., 2010.

[35] (2010, Sept.) Netfilter homepage. [Online]. Available: http://www.netfilter.org

[36] D. Sivchenko and N. Bayer, "IEEE 802.11b Standard: Maximal Throughput of UDP Traffic with different packet sizes," Deutsche Telekom Laboratories, Tech. Rep., 2004.

[37] S. Garg and M. Kappes, "Can I add a VoIP call?" *Proceedings of the IEEE International Conference on Communications*, vol. 2, pp. 779–783, May 2003.

[38] 3GPP TS 22.105, *Technical Specification Group Services and System Aspects Service aspects; Services and service capabilities*, 3GPP Technical Specification 22.105, 1999.

[39] (2010) MadWifi. The Madwifi project. [Online]. Available: http://madwifi-project.org/

[40] (2010) Iperf. NLANR applications support. [Online]. Available: http://dast.nlanr.net/Projects/Iperf/

[41] (2010) TCPDUMP. tcpdump/libpcap. [Online]. Available: http://www.tcpdump.org/

[42] (2010) WIRESHARK. [Online]. Available: http://www.tcpdump.org/

Appendix A

Budget

In this appendix we present the budget which is necessary to achieve the goals of the project: analyzing the performance of SoftToken and developing the necessary functionalities to integrate it into the CARMEN framework.
First, we list the material that it is used; then, we identify the phases of the project and we plot its evolution in time by using a Gantt chart; finally we compute the total cost of the project.

A.1 Material

The material which is needed in this project consists of:

- 3 laptops Fujitsu-Siemens Lifebook S7020 equipped with an Intel Pentium M processor at 1.73 GHz and 1 GB of RAM memory, running Ubuntu 9.10. These are used for the testbeds.
- 3 Lancom Air Lancer MC-54ag wireless PCMCIA cards with chipset Atheros which are also employed in the testbeds.
- 1 laptop Fujitsu Lifebook S6420, running Windows XP which is employed for documentation.

A.2 Project Phases

In this section we identify the main phases in which this project can be splitted. Later, we will assign an effort to each of them to quantify the cost.
The main phases of the project are:

- **Phase 1: Studying the State of the Art**
  - studying the modules of the CARMEN Project which are related with SoftToken. These modules comprise the Deliverables released by the WP1 as well as the WP2 which are related to the CARMEN architecture and the media access control mechanisms respectively.
  - studying the characteristics of WMNs.
  - studying IEEE standards 802.11, 802.16.
- **Phase 2: Studying the SoftToken Mechanism**
  - studying and understanding the source code of SoftToken.
  - learning how to run SoftToken on a testbed.
  - running simple tests in debug mode and tracing the messages.
  - generating flow diagrams.
- **Phase 3: Developing a Model for IEEE 802.11 and for SoftToken**
  - studying related papers.
  - building the models.
  - computing the results for the models.
- **Phase 4: Tests and Validation**
  - designing the tests.
  - running the tests.
  - processing the results.
  - validating the results.
  - editing the source code.
  - setting up demonstrations to show the behaviour of SoftToken in real-time.
- **Phase 5: Implementation of New Functionalities**
  - identifying the requirements.
  - designing the new functionalities.
  - implementing the functionalities.
  - debugging the functionalities.
- **Phase 6: Documentation**
  - generating periodical reports describing the work which is done.
  - assisting to periodical syncronization meetings to update the status of the project and setting new milestones.
  - generating presentations describing the evolution of the work.
  - merging the previous reports into this thesis.

We consider that the project lasts 7 months from February to August 2010, even if the internship lasts 8 months, because the last month of work is not included in the thesis. We consider a working day of 8 hours and a working week of 5 days.
The Gantt chart plotted in Figure A.1, shows the evolution of the phases of the project.

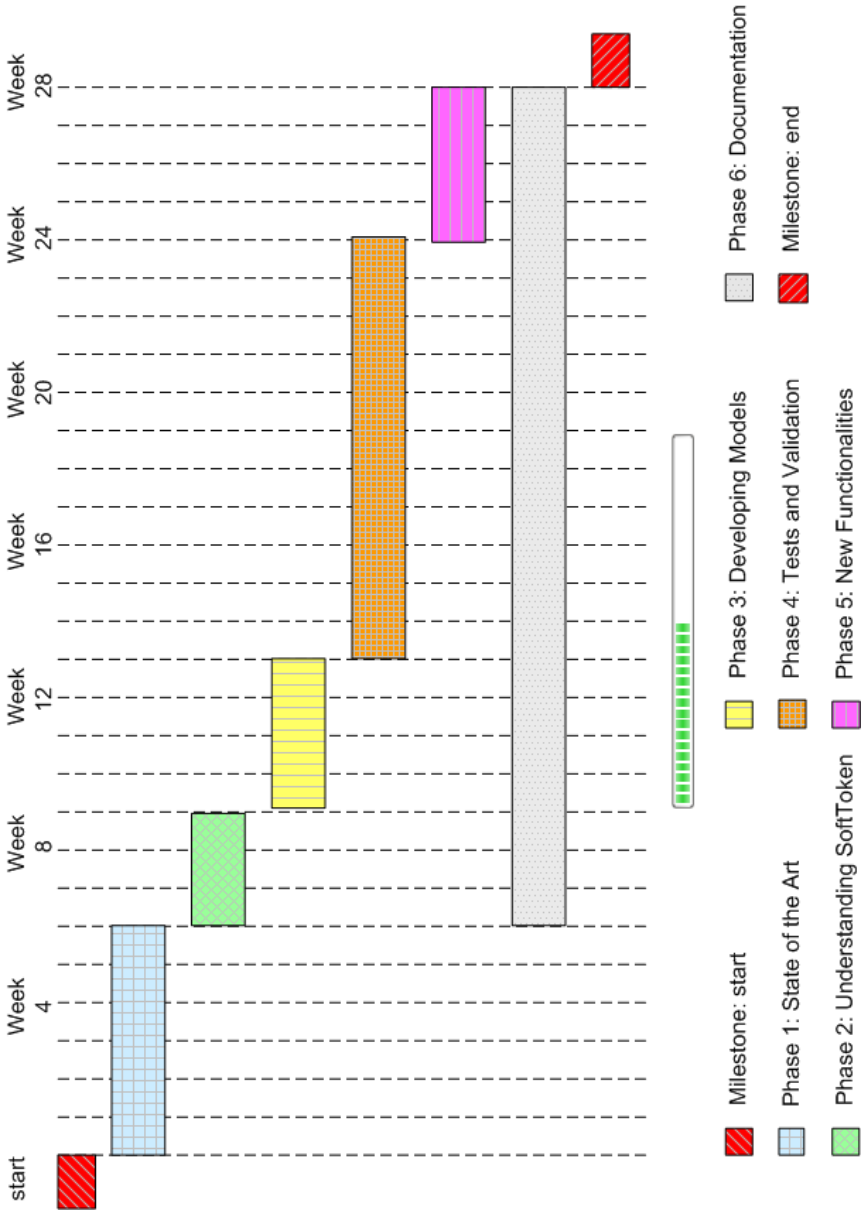


Figure A.1: Scheduling of the project

A.3 Material Expenses

Material expenses are showed in Table A.1.

Concept	Unities	Cost/unit	Price (Euros)
Fujitsu-Siemens Lifebook S7020	3	700	2100
Fujitsu Lifebook S6420	1	700	700
Lancom Air Lancer wireless PCMCIA cards	3	60	180
Total without V.A.T.	-	-	2980
V.A.T. (16%)	-	-	476.8
Total with V.A.T.	-	-	3456.8

Table A.1: Material Expenses

A.4 Human Resources Expenses

In this section we compute the wages perceived by the telecommunications engineer in exchange for his labour in the project. We will consider an average wage of 50 Euros per hour of work.
In order to quantify the labour performed by the engineer we assign an effort to each of the phases described in section A.2, as we said considering a working day of 8 hours and working week of 5 days. Table A.2 shows the labour expenses for the project.

Concept	Hours	Effort	Total amount (Euros)
Studying the State of the Art	240	100%	12000
Studying the SoftToken mechanism	120	80%	4800
Developing Models	160	80%	6400
Tests and Validation	440	80%	17600
New Functionalities	160	80%	6400
Documentation	880	20%	8800
Total	$240 + 0.8 * 880 + 0.2 * 880 = 1120$	-	56000

Table A.2: Human resources expenses

A.5 Total Expenses

Adding together both the material expenses and the labour expenses we obtain the total budget for the project which is shown in Table A.3.

Concept	**Cost (Euros)**
Material Resources (V.A.T. included)	3456.8
Human Resources	56000
Total	59456.8

Table A.3: Total budget for the project

The total budget for the project is fifty nine thousand, four hundred and fifty six Euros and eight cents.

Appendix B

Resumen en Español

Las Redes de Malla Inalámbricas (WMN en inglés) permiten integrar a un coste razonable otros tipos de redes, haciendo de ellas las candidatas idóneas para facilitar el despliegue eficiente y flexible de Redes de Siguiente Generación (NGN en inglés).
La amplia presencia de las Redes de Area Local Inalámbricas (WLAN en inglés) basadas en el estándar IEEE 802.11 explica el que la mayoría de las WMNs actualmente desplegadas estén basadas en dicho estándar. Sin embargo, la naturaleza distribuida de IEEE 802.11 crea problemas en las redes multisalto debido al mecanismo de acceso al medio utilizado (CSMA/CA) que genera colisiones disminuyendo la eficiencia y la calidad del servicio.
En este Proyecto Fin de Carrera, presentamos el análisis del rendimiento de SoftToken, que propone un novedoso Mecanismo de Acceso al Medio (MAC en inglés) coordinado, basado en el principio del *token* o testigo. El objetivo de SoftToken consiste en añadir un mecanismo de paso de testigo por encima del estándar IEEE 802.11 que evite las colisiones. También se han implementado algunas funciones para integrar este protocolo en la arquitectura diseñada por el proyecto CARMEN (CARrier grade MEsh Networks).

B.1 Introducción

En este Proyecto Fin de Carrera presentamos el trabajo realizado durante las prácticas realizadas en el departamento de Seamless Communications de los Deutsche Telekom Laboratories en Berlin de Febrero a Septiembre de 2010.

B.1.1 Motivación

Los avances en las comunicaciones inalámbricas de las últimas décadas han transformado nuestro día a día. Las tecnologías inalámbricas como la telefonía móvil, las redes de área local

inalámbricas WiFi o el WiMAX están evolucionando para ofrecer banda ancha móvil y ubicua a los usuarios.
Por un lado, los sistemas de telefonía móvil 3G que ya ofrecían cobertura mundial, evolucionan hacia la cuarta generación 4G como la Evolución a Largo Término (LTE en inglés) para alcanzar anchos de banda superiores a los 300 Mbps en el enlace de bajada o a los 75 Mbps en el enlace de subida. Por otro, la tecnología WiFi que está ampliamente extendida ofrecerá anchos de banda de hasta 540 Mbps con el nuevo estándar IEEE 802.11e. En los próximos años también se espera que incremente el despliegue de WiMAX en zonas urbanas.
Las redes inalámbricas de área personal (WPAN en inglés) así como las redes de sensores inalámbricas (WSN en inglés) también están cobrando creciente importancia en campos como el control y la monitorización remotos, los dispositivos periféricos o la domótica.
Para ofrecer disponibilidad ininterrumpida de datos, voz y tráfico multimedia, es necesario integrar estas múltiples redes. En este sentido, las redes móviles de siguiente generación (NGMN en inglés) representan la evolución en la arquitectura de redes hacia el todo-IP para las redes de acceso.
Las Redes de Malla Inalámbricas (WMN en inglés) pueden integrar a un coste razonable otros tipos de redes, convirtiéndolas en las candidatas idóneas para facilitar el despliegue eficiente y flexible de Redes de Siguiente Generación (NGN en inglés). Sin embargo, el diseño de este tipo de redes plantea una serie de desafíos como son la escalabilidad, el soporte para la calidad de servicio, la autoconfiguración o la seguridad. Esto constituye la motivación para el proyecto europeo CARMEN cuyo objetivo es ofrecer servicios con grado de operadora en WMN heterogéneas a un coste razonable. Para ello, se ha diseñado y desarrollado una arquitectura completa para este tipo de redes.
Debido al amplio despliegue de IEEE 802.11, la mayoría de WMN está basada en este estándar que utiliza CSMA/CA como técnica de acceso al medio. Este mecanismo de contienda no es escalable en redes multisalto debido al incremento de colisiones que se produce cuando aumenta el número de nodos y que disminuye la eficiencia. Añadiendo un mecanismo de acceso coordinado por encima de IEEE 802.11 es posible controlar el acceso al medio evitando las colisiones y mejorando el rendimiento. En este Proyecto Fin de Carrera analizamos un nuevo protocolo llamado SoftToken que ha sido desarrollado por los T-Labs para el proyecto CARMEN. SoftToken implementa un mecanismo de acceso al medio coordinado para WMNs basadas en IEEE 802.11 evitando las colisiones y ofreciendo soporte para calidad de servicio.

B.1.2 Objetivos y contribución

Los principales objetivos de este Proyecto Fin de Carrera han sido: estudiar, caracterizar, depurar el código y desarrollar nuevas funcionalidades para SoftToken. Las contribuciones de este Proyecto Fin de Carrera han sido:

- Estudiar el rendimiento de SoftToken tanto de forma teórica como experimental y compararlo con el de IEEE 802.11 que opera por debajo. Para ello, primero hemos estudiado el rendimiento de IEEE 802.11 y, empleando el mismo tipo de análisis hemos propuesto un modelo para SoftToken.

- Durante la fase de caracterización han aparecido algunos errores en la implementación que pueden afectar al rendimiento y hemos tenido que resolver.
- Al tiempo que íbamos depurando el código, se han realizado diferentes pruebas con distintos escenarios y configuraciones para validar los modelos y hallar los valores del rendimiento de SoftToken.
- Por último implementar un nuevo módulo para SoftToken para integrarlo en la arquitectura de CARMEN y permitir que algunas primitivas modifiquen los parámetros de configuración del planificador de SoftToken. Dichos parámetros determinan la Calidad del Servicio (QoS en inglés) en términos de ancho de banda y prioridad del tráfico proporcionada por SoftToken.

B.2 El proyecto CARMEN

El proyecto CARMEN (CARrier grade MEsh Networks en inglés) del séptimo programa Marco de la Unión Europea es un proyecto de tres años de duración: de enero de 2008 a diciembre de 2010. CARMEN está compuesto por ocho entidades que incluyen dos operadoras: Deutsche Telekom y BT; dos fabricantes: NEC y Alcatel Lucent; un instituto de investigación: Fraunhofer FOKUS; y tres universidades: University College Dublin, Universidad Carlos III de Madrid y AGH University of Science and Technology de Cracovia. El proyecto CARMEN tiene como objetivo especificar y desarrollar la arquitectura de una WMN que esté compuesta por distintas tecnologías inalámbricas y que soporte servicios con grado de operador [2].

B.2.1 Objetivos de CARMEN

El objetivo principal de CARMEN es mejorar la eficiencia y la escalabilidad en las WMNs que constituyen dos de las principales limitaciones en este tipo de redes. Alguno de los objetivos específicos de CARMEN son:

- Proveer servicios con grado de operador en WMNs ofreciendo una calidad tan cercana a la ofrecida en redes cableadas como sea posible.
- Soportar múltiples tecnologías de acceso inalámbricas proporcionando una abstracción de las capas MAC para las distintas tecnologías como pueden ser IEEE 802.11, IEEE 802.16 o Digital Video Broadcasting (DVB).
- Emplear eficientemente el espectro radioeléctrico mejorando las funcionalidades de la capa MAC y soportar planificaciones de red dinámicas.
- Ofrecer soporte para movilidad ampliando el estándar IEEE 802.21 que describe la arquitectura para traspasos entre familias 802 de forma independiente (MIH en inglés) para que soporte funcionalidades adicionales de gestión en WMN.

- Proveer servicios broadcast y multicast como DVB o TV móvil.
- Soportar mecanismos de autoconfiguración y monitorización para reducir los costes de operación y gestión, y soportar cambios dinámicos de la topología.

B.2.2 Arquitectura

CARMEN se ha diseñado para hacer frente a dos escenarios que representan dos casos de uso con distintas funcionalidades: WMNs para entornos urbanos y WMNs para casos de emergencia. Un ejemplo de una red CARMEN típica se muestra en la Figura 3.1 en la cual existe una conexión entre un terminal de usuario (UT en inglés) de la red CARMEN y un terminal WiMAX a través de Internet. En dicha Figura se describen los distintos tipos de nodos definidos en CARMEN.
El diseño de una WMN formada por múltiples tecnologías de acceso inalámbricas requiere la abstracción de dichas tecnologías para poder integrarlas de forma homogénea. El primer nivel de abstracción corresponde a los enlaces definidos en CARMEN. La abstracción de los recursos se lleva a cabo transformando enlaces físicos inalámbricos en enlaces lógicos (LL en inglés) los cuales proveen únicamente propiedades genéricas como ancho de banda o SNR. Los enlaces lógicos que forman un dominio de colisión se agrupan en grupos de enlaces (LG en inglés). En cada LG, uno de sus nodos denominado controlador del LG (LGC en inglés) gestiona el LG y el resto de nodos se denominan agentes de enlace (LA en inglés). Por último, los pipes o tuberías en las cuales los recursos han sido reservados permiten la comunicación a nivel 3. La Figura 3.2 muestra un ejemplo del modelo de abstracción de enlaces en CARMEN. Los LGCs son específicos para cada interfaz inalámbrica, es decir, dependen de la tecnología inalámbrica empleada. Para permitir que las capas superiores empleen un único conjunto de primitivas para todas las tecnologías subyacentes, CARMEN define dos componentes básicos que se muestran en la Figura 3.3:

- La Función de Gestión de la Interfaz (IMF en inglés) que emplea las primitivas de IEEE 802.21 siempre que sea posible y define otras para la gestión de la WMN.
- Los Adaptadores MAC (MAd en inglés) que proveen la abstracción de los recursos y los mecanismos de gestión del LG. Están compuestos por el módulo correspondiente al LGC o al LA así como por el módulo de monitorización (MoMa en inglés).

La Figura 3.3 también muestra los planos de datos y control en un nodo perteneciente a CARMEN. En el plano de control se encuentran las llamadas funciones de malla entre las cuales están las funciones de enrutamiento y autoconfiguración. Éstas operan por encima de la capa de abstracción formada por la IMF que provee de un conjunto de primitivas a los módulos que están por encima y por debajo. El plano de datos depende de la tecnología de acceso al medio subyacente y está compuesto por el módulo MPLS de reenvío de paquetes que es el responsable de gestionar el tráfico de nivel de red en CARMEN.

B.2.3 Módulos y requerimientos funcionales

Los distintos módulos definidos en CARMEN son:

- **Autoconfiguración (SCF)** cuyos objetivos son: el descubrimiento de nodos, la formación inicial de la topología, y la adaptación y optimización durante el funcionamiento normal de la red así como durante los fallos, de forma autónoma.
- **Monitorización** cuyo objetivo es proporcionar a tiempo información precisa a los otros módulos.
- **Manejo de la capacidad** cuyas funciones son: el control de admisión, que comprueba si se acepta o se rechaza una nueva conexión, el inicio y mantenimiento de las tuberías o pipes, y la función de control de poliza que comprueba que un flujo de datos cumple con los requisitos establecidos en términos de ancho de banda y clase de tráfico.
- **Enrutamiento (RtF)** provee conectividad entre nodos CARMEN. Sus funciones principales son: el descubrimiento de la topología de red y la diseminación de esta información, la gestión de la capacidad y el reenvío mediante MPLS.
- **IMF** proporciona un conjunto de primitivas de interfaz abstracta (AI en inglés) para las funciones de capas superiores de CARMEN mencionadas previamente así como para los MAds de capas inferiores. Las primitivas que se han empleado en este Proyecto Fin de Carrera corresponden a aquellas responsables de la configuración inicial de un LG y de la gestión de recursos en dicho LG. La estructura de estas primitivas está especificada en la sección 3.5.5.
- **Extensiones y Adaptadores MAC** proporcionan los mecanismos necesarios para comunicar los módulos de CARMEN de capas superiores que son independientes de la tecnología con las distintas tecnologías inalámbricas. En un nodo CARMEN existe, por cada interfaz inalámbrica, un MAd que la gestiona. En la fase de arranque, el MAd es el responsable de fijar la configuración inicial de la interfaz inalámbrica y de arrancar la función de autoconfiguración. El módulo de monitorización le proporciona una lista con los nodos vecinos y con esta información decide qué nodos pertenecen al mismo LG eligiendo un único LGC. La arquitectura de un MAd se muestra en la figura 3.4. En CARMEN se definen dos tipos de MAds para IEEE 802.11, el descoordinado y el coordinado. En este Proyecto Fin de Carrera nos hemos centrado en SoftToken que constituye una extensión MAC coordinada basada en testigo y lo hemos comparado con SoftTDMAC que constituye otra extensión MAC para IEEE 802.11 basada en TDMA.

B.3 El protocolo SoftToken

SoftToken es un protocolo MAC coordinado basado en testigo para IEEE 802.11 cuyo objetivo es evitar las colisiones y garantizar la calidad de servicio.

SoftToken opera en un LG en el cual el LGC funciona como maestro y representa la entidad central que coordina la transmisión de los demás LAs del LG que funcionan como esclavos. Dicha coordinación de la transmisión está basada en el principio del testigo para evitar colisiones. De esta manera, únicamente puede transmitir el nodo que posee el testigo. Regularmente, el maestro – que contiene la lista de vecinos con las direcciones MAC de los nodos esclavo del LG – envía el testigo mediante un paquete llamado *token request* a un esclavo cada vez, que indica la cantidad y el tipo de tráfico que puede ser transmitido por el esclavo antes de devolverle el testigo con un paquete llamado *token response*. En este paquete, el esclavo indica cuánto tráfico ha sido transmitido así como el tráfico que espera en sus colas para ser enviado. Después de recibir el *token response*, el maestro sirve su necesidad de comunicación transmitiendo la cantidad indicada por el planificador o *scheduler* antes de volver a transmitir un *token request* al siguiente esclavo. Este mecanismo para tres nodos se muestra en la figura 4.1 que muestra el intercambio de mensajes entre los nodos.

B.3.1 Implementación

SoftToken está implementado tanto en espacio usuario como en espacio núcleo como se muestra en la Figura 4.9 y puede operar en modo maestro o en modo esclavo. El espacio usuario está formado por el *SoftToken Controller* que se emplea para configurar y monitorizar el espacio núcleo.

B.3.1.1 Implementación del espacio núcleo

Los principales módulos del espacio núcleo son el cliente SoftToken, el coordinador SoftToken y el planificador. Todos los nodos que ejecutan SoftToken en un LG tienen la funcionalidad del cliente pero únicamente el nodo que ejerce de LGC, tiene, adicionalmente, la funcionalidad de coordinador.
El módulo **cliente** controla las cuatro colas de paquetes definidas en SoftToken. En SoftToken cada cola tiene una clase de tráfico asociada con lo que hay como máximo cuatro posibles clases de tráfico. Los paquetes IP salientes son encolados por SoftToken en la cola que tenga asociada aquella clase de tráfico que coincida con el valor del tipo de servicio de la cabecera IP del paquete. En caso de que no haya coincidencia entre la clase de tráfico y el tipo de servicio, los paquetes son encolados por defecto en la cola 0. El uso de estas colas junto con el planificador se emplea en SoftToken para permitir la diferenciación de tráfico proveyendo distintas calidades de servicio.
El modo esclavo es el modo de operación por defecto en SoftToken. Cuando SoftToken se inserta en modo esclavo, únicamente implementa la funcionalidad de cliente: registra el dispositivo SoftToken en el núcleo, inicializa alguno de los atributos del nodo y se conecta al *hook* de Netfilter que controla el paso de paquetes por la pila del protocolo IPv4. En caso de que se inserte en modo maestro, además de las funcionalidades del cliente, el nodo cuenta con las funcionalidades de coordinador que cuenta con un temporizador para la retransmisión del testigo en caso de pérdida. Después de insertar un módulo SoftToken en modo maestro o esclavo

es necesario configurarlo mediante el controlador para indicarle la interfaz inalámbrica que va a emplear SoftToken. Estas operaciones se muestran en el diagrama de la Figura 4.10.
Una vez que la interfaz ha sido configurada, un nodo esclavo espera a que el *hook* atrape algún paquete que puede ser tanto entrante como saliente. Si un paquete saliente va destinado a la interfaz que emplea SoftToken, se examina el tipo de servicio de la cabecera IP del paquete y se encola en aquella que tiene la misma clase de tráfico asignada o en 0 si no corresponde con ninguna de las clases de tráfico configuradas. En caso de que el paquete vaya destinado a otra interfaz, se le deja pasar. La Figura 4.11 muestra el diagrama de flujo para esta funcionalidad.
Cuando se captura un paquete entrante, el nodo esclavo tiene que comprobar si dicho paquete contiene un mensaje SoftToken proveniente del maestro para procesarlo o por el contrario se trata de otro tipo de paquete en cuyo caso se le permite el paso. Cuando un nodo esclavo recibe un mensaje *softtoken_mgnt_request* del maestro (véase Figura 4.12), SoftToken libera de cada cola tantos paquetes o bytes como indica dicho mensaje, transmitiendo los paquetes a sus correspondientes destinos. A continuación, genera un mensaje *softtoken_mgnt_response* para el maestro, lo transmite y espera a que se atrape otro paquete (véase Figura 4.13).
El módulo **coordinador** es el responsable de la coordinación del testigo. Añade un ciertas funcionalidades adicionales a las del cliente como son un temporizador que decide cuándo se ha perdido el testigo o la asignación de recursos configurada.
Cuando SoftToken se inserta en modo maestro, después de realizar las tareas del cliente, inicializa el temporizador y le asocia la función que tiene que llamar cuando expira. Hasta que no se añada un nodo esclavo en el LG y se configure en el maestro, éste será el único nodo que será planificado. Por lo tanto, cuando expira el temporizador el maestro se planifica como el siguiente nodo a ser requerido por él mismo. Genera el *softtoken_mgnt_request*, se lo envía y pone el temporizador a 2 segundos. Como dicho mensaje tiene como origen y destino el mismo nodo, no se transmite por la interfaz inalámbrica. A continuación se atrapa dicho paquete y, como se ha explicado para el caso del módulo cliente, comprueba el tipo de mensaje, lo reconoce, libera los paquetes correspondientes y genera el mensaje *softtoken_mgnt_response* para si mismo. Este paquete a su vez es atrapado y reconocido por SoftToken que actualiza las estadísticas para los paquetes que ha liberado anteriormente y que siguen actualmente en sus colas. En ese momento, fija el valor del temporizador para que expire inmediatamente, volviendo a empezar el ciclo descrito en este párrafo. Este mecanismo se muestra en la Figura 4.14.
Cuando un esclavo se añade a la lista de vecinos, después de planificarse a si mismo, el maestro selecciona el siguiente esclavo como el nodo actual al que pasarle el testigo. Genera el mensaje *softtoken_mgnt_request* para el esclavo que en este caso sí es transmitido por la interfaz inalámbrica y fija el temporizador a dos segundos que se corresponde con el valor del temporizador para retransmisiones. Si después de dos segundos no se ha recibido el paquete *softtoken_mgnt_response* del nodo solicitado, el temporizador expira, asumiendo el maestro que el testigo se ha perdido, reiniciándose el ciclo de planificación. Si el paquete *softtoken_mgnt_response* se recibe antes de que expire el temporizador, el maestro actualiza las estadísticas para los paquetes transmitidos y encolados en el esclavo correspondiente y fija el valor del temporizador para que expire inmediatamente.
La última posibilidad que se puede dar ocurre cuando el paquete *softtoken_mgnt_response* se recibe después de que el maestro haya asumido que el testigo se ha perdido. Esto ocurre durante periodos transitorios en la fase inicial de SoftToken hasta que los mensajes se han sincronizado. Al comprobar el paquete *softtoken_mgnt_response* retrasado, el maestro reconoce

que corresponde a un nodo que estaba siendo requerido anteriormente y lo planifica como el siguiente al cual pasarle el testigo.
La configuración del **scheduler** determina la calidad de servicio en términos de ancho de banda y clase de tráfico que están asignadas a cada cola. Esto permite garantizar diferentes anchos de banda para distintos tipos de tráfico. En la versión de SoftToken que presentamos, el scheduler se configura manualmente en el maestro utilizando los comandos proporcionados por el módulo controlador. En la siguiente sección describimos la funcionalidad desarrollada para permitir que alguna de las primitivas de CARMEN modifique dinámicamente la configuración del scheduler como respuesta a un evento de gestión de recursos como puede ser una petición, modificación o liberación. La configuración del scheduler se almacena en el nodo maestro. La información almacenada para cada una de las colas de SoftToken es:

- El número de colas
- El tipo de recurso asignado a una cola, que pueden ser: paquetes, bytes o tiempo.
- La cantidad de recursos asignados a la cola
- Un retardo que indica el tiempo que el planificador tarda en planificar el siguiente nodo.
- La clase de tráfico asignada a la cola

La configuración del scheduler es señalizada por el maestro a los esclavos empleando mensajes *softtoken_mgnt_request*. De acuerdo con esta configuración, los nodos liberan por cada cola el número de paquetes o bytes indicado en dichos mensajes.

B.3.1.2 Implementación del espacio usuario

El espacio usuario está formado por el módulo controlador que se emplea para monitorizar, controlar y configurar el espacio kernel de SoftToken. Para ello se emplean ioctls. Algunas de las funcionalidades del controlador son:

- Fijar la interfaz inalámbrica que va a ser empleada por SoftToken
- Obtener la interfaz que está siendo empleada por SoftToken y la configuración del planificador (esto último, sólo en modo maestro)
- Añadir y eliminar esclavos, sólo en modo maestro
- Asignar recursos para una clase de tráfico, sólo en modo maestro
- Fijar el tiempo que el planificador espera para planificar el siguiente nodo, sólo en modo maestro
- Mostrar una lista con el modo de funcionamiento, las direcciones MAC e IP, y el número de paquetes y bytes actualmente encolados

B.3.2 Contribución

La versión de SoftToken que se ha presentado sólo puede configurarse manualmente empleando el módulo controlador del espacio usuario. Para poder integrar SoftToken en el sistema diseñado por CARMEN, es necesario proveer una interfaz al MAd coordinado, de forma que ciertas primitivas definidas en el IMF puedan controlar los parámetros de SoftToken. Hemos desarrollado las extensiones necesarias en SoftToken para permitir dicha funcionalidad.

B.3.2.1 El entorno CARMEN y la integración de SoftToken

En primer lugar hemos modificado tanto el MAd coordinado como SoftToken para permitir que la SCF configure un LG coordinado por SoftToken. Para ello, la SCF tiene que lanzar SoftToken en modo maestro en el LGC y configurar la lista de direcciones MAC del resto de vecinos, y lanzar SoftToken en modo esclavo en el resto de nodos del LG. En segundo lugar, hemos introducido las modificaciones necesarias para que una vez que la SCF ha configurado SoftToken en el LG, se permita a la RtF asignar recursos para distintas clases de tráfico.
La Figura 4.15 muestra como se integra SoftToken en el sistema diseñado para CARMEN. Las partes coloreadas representan aquellos módulos que han sido modificados o añadidos para integrar SoftToken. Las líneas magenta y turquesa representan el flujo de información para una petición de asignación de recursos en modo maestro; rojas y azules para una en modo esclavo. Para ello, el manejador en el MAd coordinado ejecuta las llamadas al sistema para insertar SoftToken en modo maestro o esclavo e indicarle la interfaz inalámbrica que tiene que emplear. En el modo maestro, la SCF proporciona una lista con las direcciones MAC de los nodos pertenecientes al LG. Si no ocurre ningún error, la primitiva devuelve un valor de éxito, en caso contrario, devuelve uno de fallo. Después de este paso, SoftToken está configurado y se está ejecutando en el LG. A continuación se han añadido las funciones necesarias para la asignación de recursos. Para gestionar la asignación de recursos: realizar una nueva asignación, modificarla o liberarla, se han añadido dos módulos en la MAC coordinada:

- **Schedulerhelper** que le pasa a SoftToken los valores de una asignación de recursos
- **setConfig** contiene el algoritmo que ajusta la configuración del planificador dinámicamente teniendo en cuenta las asignaciones que existen en el planificador

Cuando se genera en el RtF de un nodo esclavo una petición de asignación de recursos es necesarion añadir a SoftToken un mecanismo que permita señalizar dicha petición al maestro así como devolver el resultado de dicha petición. Para ello, hemos modificado el formato de los paquetes de gestión empleados por SoftToken incluyendo la información correspondiente a la petición y a la respuesta, e introduciendo los módulos necesarios para manejarla.
El algoritmo de configuración del planificador está implementado en la función setConfig que se llama cuando tiene lugar una asignación de recursos. Dicho algoritmo tiene en cuenta: la asignación de recursos actual, la clase de tráfico de la petición de asignación y el valor de dicha petición. En base a esto, el algoritmo garantiza una asignación mínima para cada clase de

tráfico en términos de ancho de banda. En primer lugar, el algoritmo asume que el ancho de banda total que se puede repartir entre todas las clases de tráfico es constante y corresponde con el ancho de banda de saturación del canal. A continuación, reparte el ancho de banda basado en la idea del *water filling*, tratando de realizar las asignaciones completas cuando es posible, y, en caso contrario, garantizando primero aquellas con una clase de tráfico con mayor prioridad. La Figura 4.16 muestra el diagrama de flujo del algoritmo, en el que se calcula el número de clases de tráfico que están actualmente en uso y después se calcula en cuál de los 16 posibles casos nos encontramos dependiendo de la clase de tráfico de la petición entrante. Para cada uno de esos casos, el algoritmo realiza distintas asignaciones basadas en *water filling*.

B.4 Análisis de SoftToken

SoftToken opera por encima de IEEE 802.11. El análisis teórico del rendimiento de UDP en IEEE 802.11 es necesario para poder evaluar el mismo rendimiento empleando SoftToken.
Debido a la naturaleza distribuida de IEEE 802.11 no es posible proporcionar un modelo determinista. Sin embargo, para un escenario con dos estaciones se pueden proporcionar resultados razonables asumiendo ciertas simplificaciones. En la sección 5.2 se ha realizado el estudio del rendimiento de UDP sobre IEEE 802.11a operando en modo DCF en un escenario de dos nodos en el cual únicamente uno de ellos envía datagramas UDP al otro. Las Tablas 5.2 y 5.3 muestran los resultados de la eficiencia de UDP para canales a 6, 36 y 54 Mbps y distintos tamaños de datagrama para DCF en modo *two-way handshake* y DCF en modo *four-way handshake* respectivamente.
Una vez obtenido los valores para IEEE 802.11a, se ha realizado el análisis teórico en el mismo escenario empleando SoftToken, el cual se ha denominado *baseline test*. En él, se considera que únicamente se transmite un paquete por la cola 0 del esclavo al maestro. Este escenario representa el peor caso para SoftToken, ya que por cada paquete enviado hay que esperar a devolver el testigo y volver a recibirlo para transmitir el siguiente. De esta manera, podemos calcular la sobrecarga introducida por SoftToken sin más que comparar los resultados con los obtenidos empleando sólo IEEE 802.11a. Los valores de la eficiencia para el *baseline test* se muestran en la Tabla 5.5 y en las Tablas 5.6 y 5.7 se muestra la contribución de cada una de las componentes en el tiempo total de transmisión entre dos tramas consecutivas en SoftToken para datagramas UDP de 20 y 1470 Bytes respectivamente.
Hemos querido aplicar el *baseline test* a una aplicación con típicos requerimientos de calidad de servicio como es la voz sobre IP (VoIP en inglés). En el apartado 5.3.3 se ha calculado el número de llamadas VoIP unidireccionales que pueden soportarse en el baseline test empleando IEEE 802.11a y SoftToken.

B.5 Pruebas y Validación

Para validar el modelo teórico presentado en la sección anterior se han realizado distintas pruebas para escenarios formados por dos nodos transmitiendo tráfico UDP y TCP de forma unidireccional o bidireccional y para una o dos clases de tráfico.

B.5.1 Pruebas UDP

B.5.1.1 Pruebas con tráfico unidireccional para SoftToken, SoftTDMAC e IEEE 802.11a

En esta sección se presentan los resultados obtenidos para las pruebas realizadas con SoftToken, SoftTDMAC e IEEE 802.11a en un escenario en el que se transmite únicamente tráfico UDP unidireccional del esclavo al maestro. Estos tests fueron realizados en el mismo lugar y alternativamente en el tiempo para someterlos a las mismas interferencias y obtener resultados comparables.
Las configuraciones empleadas en las pruebas de SoftTDMAC para la asignación de recursos han sido:

- Configuración 20%M-70%S. Los *slots* de las tramas están repartidos como: 10% control, 20% maestro, 70% esclavo
- Configuración 45%M-45%S. Los *slots* de las tramas están repartidos como: 10% control, 45% maestro, 45% esclavo
- Configuración 70%M-20%S. Los *slots* de las tramas están repartidos como: 10% control, 70% maestro, 20% esclavo

Las tres configuraciones para la asignación de recursos en SoftToken han sido:

- 8 paquetes o 11,760 bytes para la cola 0, ningún paquete para el resto
- 16 paquetes o 23,520 bytes para la cola 0, ningún paquete para el resto
- 32 paquetes o 47,040 bytes para la cola 0, ningún paquete para el resto

Hemos empleado únicamente una clase de tráfico con ToS=0. Cada test UDP ha tenido una duración de 100 segundos, obteniendo estadísticas acerca del ancho de banda efectivo, *jitter* y pérdida de paquetes cada 10 segundos. Con los resultados obtenidos para SoftToken e IEEE 802.11a hemos calculado la media y la desviación estándar para los valores obtenidos. Los correspondientes al ancho de banda efectivo empleando SoftTDMAC se muestran en la Tabla 6.2, los obtenidos para SoftToken e IEEE 802.11 junto con sus errores se muestran en la Tabla 6.3. Para poder comparar estos resultados se muestran en la Figura 6.7. Lo mismo se ha realizado para los valores obtenidos para el *jitter* y la pérdida de paquetes. Respecto al ancho de banda efectivo se aprecia que SoftTDMAC tiene el más bajo, viniendo a continuación SoftToken, para el cual, como era de esperar, aumenta a medida que aumenta el número de paquetes asignado a la cola y la tasa de envío del canal. Finalmente se muestra IEEE 802.11a que proporciona los mejores valores.
Para el caso del *jitter* en cambio, SoftTDMAC proporciona los valores más bajos que se explican por estar basado éste en TDMA que proporciona estrictos valores de temporización. En general, los valores del *jitter* disminuyen a medida que aumenta la tasa de canal debido a que

los paquetes esperan durante menos tiempo en cola antes de ser transmitidos.
En el caso de la pérdida de paquetes, se mantiene por debajo del 1% excepto para IEEE 802.11a a una tasa de canal de 54 Mbps para la cual alcanza un valor superior al 8% debido probablemente a que se está superando la capacidad del canal.

B.5.1.2 Pruebas con tráfico bidireccional para SoftToken e IEEE 802.11a

El la sección 6.1.3 se muestran los resultados obtenidos para las pruebas realizadas con SoftToken e IEEE 802.11a en un escenario en el que los dos nodos transmiten una única clase de tráfico UDP para distintas configuraciones de SoftToken a una tasa de canal de 54 Mbps. El objetivo de esta prueba es mostrar la evolución del ancho de banda ofrecido y obtener el valor del ancho de banda de saturación.
La Figura 6.11 muestra los resultados obtenidos para el ancho de banda y su error para cinco configuraciones distintas de SoftToken e IEEE 802.11a.
Se puede observar que el mayor ancho de banda de saturación corresponde a IEEE 802.11a y está alrededor de los 15 Mbps. Para SoftToken, éste depende del número de paquetes configurados para la cola 0, alcanzando un ancho de banda de saturación absoluto entorno a los 10 Mbps para 16 paquetes.

B.5.1.3 Pruebas para la diferenciación de tráfico en SoftToken e IEEE 802.11a

En la sección 6.1.4 se muestran los resultados correspondientes a las pruebas realizadas en un escenario con dos nodos en el que se transmite tráfico bidireccional de distinta prioridad. En el enlace ascendente, de esclavo a maestro, se ha configurado el tráfico con más prioridad, y en el descendente, el de menos. El objetivo de estas pruebas es mostrar que SoftToken es capaz de ofrecer diferenciación de tráfico mediante distintas configuraciones de tráfico en sus colas frente a IEEE 802.11 que no es capaz de ofrecer dicha funcionalidad. Para ello se ha fijado una relación de 8 a 1 entre el tráfico de mayor prioridad y el de menor en SoftToken, configurando 32 paquetes para la cola 3 (la de más prioridad), y 4 paquetes para la cola 0 (la de menos prioridad). Los valores obtenidos para el ancho de banda y su error, el *jitter* y las pérdidas de paquetes se muestran en las Figuras 6.17, 6.18 y 6.19 respectivamente. Se puede observar que efectivamente, SoftToken proporciona diferenciación de tráfico, sin embargo, la relación entre el ancho de banda de mayor prioridad y el de menor no es de 8 a 1 sino de 6 a 1. Esta diferencia se explica porque para esta configuración se está saturando el canal y no porque SoftToken no sea capaz de repetar la relación configurada.

B.5.2 Pruebas TCP

B.5.2.1 Pruebas con tráfico unidireccional para SoftToken, SoftTDMAC e IEEE 802.11a

En estas pruebas se obtienen los resultados de las pruebas para tráfico TCP en el mismo escenario descrito para el caso de tráfico UDP unidireccional. Los valores obtenidos son muy similares a los obtenidos en la prueba equivalente con UDP. Sin embargo, en este caso también proporcionamos un análisis de los números de secuencia de TCP obtenidos que proporciona una información interesante acerca del funcionamiento de SoftToken.
Para intentar explicar la diferencia entre el ancho de banda obtenido para IEEE 802.11a y SoftToken, mostramos los gráficos tanto para los números de secuencia de TCP frente al tiempo como para la eficiencia por paquete frente al tiempo. Esto se muestra de la Figura 6.24 al a 6.31. Como se puede observar, en las pruebas correspondientes a SoftToken se aprecian intervalos en los que la transmisión se interrumpe mientras que en el caso de IEEE 802.11a la transmisión es continua. La interrupción en la transmisión tiene una duración aproximada de 2 segundos que corresponde al valor del temporizador de retransmisión de SoftToken. Para probar que dicho intervalo corresponde efectivamente con la pérdida de un paquete de gestión de SoftToken, hemos incrementado el valor del temporizador de retransmisión de SoftToken de 2 segundos a 8 segundos, y hemos vuelto a hacer las pruebas. Los resultados para estas pruebas se muestran de la Figura 6.32 a la 6.35. Se observa claramente el aumento del intervalo en el que la transmisión está interrumpida a 8 segundos probando que dicha interrupción está causada por una pérdida de un paquete de gestión de SoftToken. Dicha pérdida periodica de los paquetes de gestión en SoftToken explica la diferencia de rendimiento tanto entre el modelo teórico propuesto y los resultados prácticos así como entre SoftToken e IEEE 802.11a.

B.5.2.2 Pruebas con tráfico bidireccional para SoftToken e IEEE 802.11a

En la sección 6.2.4 se muestran los resultados del ancho de banda para las pruebas con una sola clase de tráfico TCP bidireccional entre dos nodos. En primer lugar se muestra la evolución del ancho de banda de saturación de TCP para distintas configuraciones de SoftToken e IEEE 802.11a tanto en el enlace ascendente como en el descendente. Esto se muestra en la Figura 6.37. Tanto para SoftToken como para IEEE 802.11a se observa que el ancho de banda en el enlace ascendente es mayor que en el descendente. Así mismo, se puede observar que para este escenario, la diferencia entre el ancho de banda efectivo proporcionado por SoftToken y el proporcionado por IEEE 802.11a ha dismunuido respecto al caso del escenario unidireccional. Esto se explica por el aumento de la contención en el escenario bidireccional que genera colisiones en IEEE 802.11a dando lugar a un menor ancho de banda efectivo.
En segundo lugar se muestra el ancho de banda tanto en el enlace ascendente como en el descendente para tasas de canal de 6, 36 y 54 Mbps, para distintas configuraciones de SoftToken y para IEEE 802.11a con resultados similares a los obtenidos para el caso UDP bidireccional.

B.6 Conclusiones y Futuras Líneas de Trabajo

Este Proyecto Fin de Carrera proporciona el primer análisis para el protocolo SoftToken, tanto teórico como práctico. Se ha desarrollado un modelo para estudiar la sobrecarga introducida por SoftToken y se han realizado diversas pruebas experimentales para obtener los límites de funcionamiento reales de SoftToken y compararlos con los obtenidos para los protocolos SoftTDMAC e IEEE 802.11a.
En general, podemos afirmar que SoftToken proporciona mejores resultados para el ancho de banda efectivo que SoftTDMAC y es capaz de soportar la diferenciación de tráfico. Sin embargo, para los escenarios probados, SoftToken ofrece un ancho de banda efectivo peor que IEEE 802.11a. Esto se debe, por un lado a la poca contención que existe en dichos escenarios, y por otro, a la reducción de la eficiencia debido a la pérdida de paquetes de gestión y la posterior espera al temporizador de retransmisión. Por lo tanto, se puede mejorar la eficiencia de SoftToken encontrando un valor óptimo para el temporizador de retransmisión. Así mismo, son necesarias más pruebas incrementando tanto el número de nodos como las clases de tráfico empleadas.